НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.934

До захисту допущено
В. о. завідувача кафедри ММСА
О.Л.Тимощук
«___» _____ 2019 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 124 Системний аналіз
на тему: «Система розпізнавання емоцій в мові людини»

Виконала:
студентка ІІ курсу, групи КА-81 мп
Палій Аліса Олегівна _____

Керівник:
професор кафедри ММСА,
д.т.н., проф. Данилов В.Я. _____

Рецензент: професор кафедри
ІБ ФТІ НТУУ «КПІ ім. І. Сікорського»,
д.т.н., проф. Качинський А.Б. _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань

Студент _____

Київ
2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ММСА
О. Л. Тимощук
«___» _____ 2019 р.

ЗАВДАННЯ

на магістерську дисертацію студентці Палій Алісі Олегівні

1. Тема дисертації: «Система розпізнавання емоцій в мові людини», науковий керівник дисертації Данилов Валерій Якович, д.т.н., професор, затверджені наказом по університету від «___» _____ № _____

2. Термін подання студентом дисертації: 13 грудня 2019 р.

3. Об'єкт дослідження: розпізнавання емоцій та ознаки, що використовуються для класифікації емоцій по голосу.

4. Предмет дослідження: програмні алгоритми та акустичні ознаки, що застосовуються при розпізнаванні емоційного стану людини за її мовою.

5. Перелік завдань, які потрібно розробити:

1) дослідити сучасний стан та особливості застосування математичного моделювання та оптимізації у вирішенні проблеми розпізнавання;

2) розробити математичну модель за допомогою методів штучного інтелекту;

3) розв'язати розроблену математичну модель та на її основі створити програмний продукт;

4) пошук даних для застосування в програмі;

5) розробити стартап-проект виведення на ринок результатів дослідження;

6) розробити концептуальні висновки за результатами наукового дослідження.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

1) Графічні матеріали та таблиці теоритичної частини;

2) Приклади функціонування створеного програмного продукту;

3) Таблиці у розділі стартап-проекту.

7. Дата видачі завдання: 05 вересня 2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації
1.	Концептуальний вступ дисертації. Формулювання об’єкта, предмета, цілі, завдань, новизни, практичної значущості результатів	05.09.2019—13.09.2019
2.	Перший розділ. Огляд літературно-інформаційних джерел. Понятійно-категоріальний апарат. Характеристика об’єкта	14.09.2019—27.09.2019
3.	Другий розділ. Розробка математичної моделі для задачі розпізнавання емоцій людини за мовленням, її розв’язок	28.09.2019—18.10.2019
4.	Третій розділ. Метод побудови прихованої марківської мережі. Пошук даних	19.10.2019—15.11.2019
5.	Четвертий розділ. Стартап-проект	16.11.2019—21.11.2019
6.	Концептуальні висновки. Перспективи розвитку отриманих рішень	22.11.2019—26.11.2019

Студент А.О. Палій

Науковий керівник дисертації В.Я. Данилов

РЕФЕРАТ

Магістерська дисертація: 111 с., 17 рис., 24 табл., 1 додаток, 20 джерел.

Об'єкт дослідження — емоції людини та їх вираження в мові.

Мета роботи — аналіз та покращення методів та алгоритмів розпізнавання емоцій людини за мовою.

Методи дослідження — моделювання алгоритмів та акустичних ознак, що застосовуються при розпізнавання і класифікації емоційного стану людини за її мовою.

В роботі виконано аналіз методів обробки цифрових сигналів, проведено дослідження характеристик мовних сигналів, аналіз математичних структур, на яких базується розпізнавання аудіосигналів та сучасних методів класифікації.

Результатом роботи є аналіз ефективності сучасних підходів до вирішення задачі розпізнавання емоцій по голосу, розробка власного двоетапного модифікованого алгоритму класифікації, який виявився ефективним під час навчання моделі.

Під час роботи реалізовано програмну реалізацію запропонованого алгоритму на мові програмування Python з використанням бібліотек для аналізу даних.

РОЗПІЗНАВАННЯ, КЛАСИФІКАЦІЯ, ЕМОЦІЇ, МОВА, PYTHON, НЕЙРОННІ МЕРЕЖІ, ОЗНАКИ

ABSTRACT

Master’s thesis: 111 pp., 17 fig., 24 tab., 1 application, 20 sources.

The object of study - human emotions and their expression in speech.

The purpose of the work is to analyze and improve methods and algorithms for recognizing human speech emotions.

Research methods - modeling of algorithms and acoustic features used in recognition and classification of a person's speech emotional state.

In this work, the method of processing digital signals had been carried out, the possible characteristics of the speech signal had been investigated. The analysis of basic mathematical structures underlying speech recognition and popular modern methods of classification was completed.

The results of work carried out is an analysis the effectiveness of current approaches to solving the problem of speech emotion recognition, development of own modified two-stage classification algorithm, wich was effective in machine learning of model.

During the work, we implemented software implementation of the proposed algorithm in Python programming language using different libraries for data analysis.

RECOGNITION, CLASSIFICATION, EMOTIONS, LANGUAGE, PYTHON, NEURAL NETWORKS, FEATURES.

ЗМІСТ

ВСТУП

Завдяки широкому розповсюдженню електронної техніки в наші дні, питання взаємодії комп’ютера та людини стрімко набирає популярності. Наразі існують різні способи отримання комп’ютером зворотнього зв’язку від людини: голосові команди, рухи тіла, частота серцевих скорочень та інше. Проте всі ці види взаємодії не можуть в повній мірі розпізнати настрій користувача. А емоції та мова тісно взаємопов’язані та відіграють істотну роль у спілкуванні з людино.

Розпізнавання емоцій в мові перетворилося із рядової задачі машинного навчання на важливу компоненту взаємодії людини та комп’ютера. Розпізнавання емоцій в мові — це багатопрофільна галузь досліджень, якаостаннім часом набуває популярності. Причиною такого інтересу може бути збільшення кількості мобільних та комп’ютерних додатків за ухилом в психологію (зокрема детектор брехні) та відеоігри. Наразі розробка методів, які можуть автоматично рохпізнавати емоції людини за голосом є актуальною задачею. В майбутньому розвиток цієї сфери може дозволити вирішити ряд соціаьних та побутових проблем, зіграти суттєву роль у питаннях безпеки.

Вирішення задачі автоматичного розпізнавання мовлення і класифікація емоційного стану спікера є цікавими насамперед компаніям, що працюють над впровадженням роботизованих систем в життя людей. Також зацікавленими є компанії, які мають справу з численною кількістю клієнтів та бажають покращити свій рівень взаємодії з ними. Дуже корисним розпізнавання може бути в системах, що надають інформацію чи послуги в режимі самообслуговування. Автоматизований інтерфейс дозволить уникнути людського фактору та дозволить працювати цілодобово.

Хоча для людей розпізнавання емоційного стану співрозмовника, зазвичай, є доволі легкою задачею, для комп’ютерних систем ця постановка залишається нетривіальною, оскільки вони не мають закладеного “інтелекту” для подібного аналізу. На сьогодні існує багато систем розпізнавання мови та ідентифікації людини за голосом, але з методом розпізнавання емоційного стану, комп’ютерні системи зможуть аналізувати не лише хто і що сказав, а ще і як це було сказано.

Емоції мають свої акустичні ознаки, що якимось чином кодуються в мовному потоці. Дослідження цих ознак дозволить краще зрозуміти процес розпізнавання емоційного стану мови і те, як ефективніше його реалізувати.

З популярністю машинного навчання зростає і кількість досліджень для вирішення задачі розпізнавання емоцій мовлення. Частину проблем уже було вирішено окремими науковцями, але ряд проблем і ідей залишаються відритими до праці над ними.

Об’єктом дослідження є системи класифікації та розпізнавання цифрових даних.

Предметом дослідження є програмні алгоритми та акустичні ознаки, які застосовуються для класифікації і розпізнавання емоцій в мові людини.

Метою роботи є покращення ефективності методів класифікації та розпізнавання емоцій в мові людини за допомогою застосування різноманітних модифікацій.

Для досягнення мети було поставлено такі задачі. Спочатку дослідити процес розпізнавання звукових сигналів, потім заглибитись у порівняльний аналіз сучасних методів класифікації, реалізувати запропонований алгоритм класифікації емоцій та виявити головні недоліки методів та алгоритмів у контексті даного дослідження (зробивши висновки щодо їх ефективності, а також запропонувати ідеї щодо подальших досліджень).

Дана робота містить чотири розділи. У перший розділ присвячено постановці задачі класифікації та розпізнавання мовленневих емоцій, характеристики мови, які розглядають, та найбільш популярні типи класифікаторів. Другий розділ містить математичну формалізацію задачі класифікації емоцій, там описуються математичні структури та вводиться алгоритм для покращення точності розпізнавання. У третьому розділі описується створений програмний продукт, його архітектура, принципи використання. Наведено якісний аналіз результатів роботи існуючих класифікаторів, порівняння їх ефективності із використаним. Четвертий розділ присвячено розробці стартап-проекту, що базується на розробленій системі.

РОЗДІЛ 1 АНАЛІЗ РОЗПІЗНАВАННЯ ЕМОЦІЙ В МОВІ

1.1 Аналіз актуальності задачі розпізнавання емоційного стану людини за мовленням

Емоції відіграють значну роль у щоденних міжособистісних взаємодіях людини. Вони важливі для прийняття раціональних та розумних рішень. Вони допомагають нам співставляти та розуміти почуття інших, передавати свої почуття та надавати відгук іншим. Дослідження виявили потужну роль, яку емоції відіграють у формуванні соціальної взаємодії людини. Емоційні прояви передають значну інформацію про психічний стан людини. Це відкрило нове дослідницьке поле автоматичного розпізнавання емоцій, що має за основну ціль розпізнавання та відтворення бажаних емоцій. У більш давніх дослідженнях було розглянуто кілька основ для розпізнавання емоційних станів, таких як міміка, мова (саме текст мовлення), фізіологічні сигнали та ін.

Порівняно з багатьма іншими біологічними сигналами (наприклад, електрокардіограма) мовні сигнали, як правило, можна отримувати більш легко та економічно. Ось чому більшість дослідників зацікавлені у розпізнаванні емоцій мовлення. Система покликана розпізнати основний емоційний стан спікера з його голосу. Протягом останніх років область набирає все більший інтерес у дослідників. Існує багато варіантів застосування виявлення емоцій людей наприклад:

- роботизований інтерфейс;
- аудіоспостереження;
- веб-електронне навчання;
- комерційні програми;
- клінічні дослідження;

- розваги;
- банківська справа;
- кол-центри;
- картонні системи;
- комп’ютерні ігри.

Також розпізнавання емоцій мовлення може стати у нагоді для електронного навчання. Інформація про емоційний стан учнів може допомогти підвищити якість викладання. Наприклад, вчитель може використовувати систему розпізнавання для вирішення того, які предмети краще викладати, і для розробки стратегії управління емоціями в навчальному середовищі. Емоційний стан учня слід враховувати в класі [1].

Мова людини в різних емоційних станах може відрізнятися за багатьма показниками. Задача розпізнавання емоцій мовлення - міждисциплінарна. Від якості її реалізації може залежати також ефективність автоматизованих систем управління, сек’юріті-систем, систем, призначених для екстреного сповіщення тощо.

Розпізнавання емоційного стану за мовленням можуть використовуватись в різних областях та не повинні потребувати високої обчислювальної потужності, особливо на фоні систем розпізнавання емоцій за фото або відео.

На сьогодні, системи, що базуються на аналізі аудіосигналів, мають досить низькі показники розпізнавання. Задача автоматичного розпізнавання емоційного стану потребує ретельного вивчення багатьох питань, як-то: які саме емоції можна виділити, які характеристики треба розглядати для найбільш ефективної ідентифікації, які класифікатори краще використовувати для отримання найкращого результату.

Для покращення точності класифікації емоційного стану за мовленням, вважається перспективним підхід, який на базується на розпізнаванні статі

спікера при обчисленнях, оскільки голосові діапазони жінок та чоловіків відрізняються. Такий підхід надає можливість більш точно розпізнавати емоційний стан людини. Ця теорія не потребує надвисокої обчислювальної потужності, так як в її основі використання ознак, що призначені лише для розпізнавання емоцій.

Підходи, що застосовуються для класифікації та розпізнавання емоційного стану за мовою останнім часов досить швидко набирають популярності, а отже і розвиваються. Дослідники використовують різні класифікатори для підвищення розпізнавальної точності. Також з часом відкриваються нові характеристики аудіосигналів, що дозволяють краще розпізнвати емоційний стан спікера. Поставлена задача є дуже важливою для досягнення ефективності у взаємодії людей та автоматизованих інтелектуальних систем. Та хоч наразі існує багато різних реалізацій задачі, досі не розроблено настільки ефективно працюючого методу, що надавав би достатню точність розпізнавання [2].

1.2 Аналіз структури системи розпізнавання емоцій по голосу

Система розпізнавання емоцій (Speech Emotion Recognition System - SER) повинна автоматично ідентифікувати емоційний стан спікера. Дана система побудована на глибокому аналізі процесу генерації звукового мовенневого сигналу, процесі вилучення заданих ознак, що можуть містити інформацію щодо емоційного стану голосу людини, і застосування певних методів розпізнавання емоцій.

Система розпізнавання емоційного стану (як і інші системи розпізнавання) складається з чотирьох незамінних частин, таких як:

1. Запис вхідного сигналу;
2. Вилучення ознак;
3. Обробка класифікатором;
4. Видача результату класифікації.

Приблизна структура системи розпізнавання емоцій мовлення показана на рисунку 1.1.

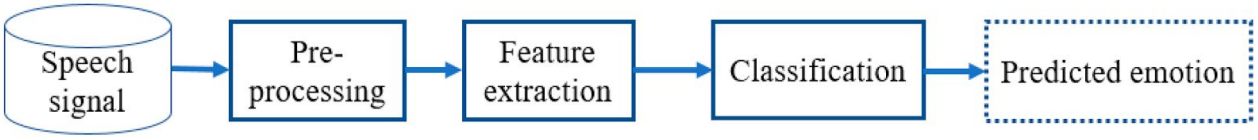

Рисунок 1.1 – Структура роботи системи розпізнавання емоцій в мові

Система розробляється з декількох етапів. Спочатку проводиться обробка сигналу, під час якої вилучаються всі величини, необхадні для роботи з сигналом (як-то тональна висота чи енергія). Потім вилучені величини потрібно просумувати в меньші набори ознак. Далі на певному датасеты навчається класифікатор, таким чином він починає пов’язувати вилучені ознаки з емоціями, які їх відповідають.

Точність роботи систем розпізнавання емоцій за мовою дуже залежить від якості даних, на яких працює система. Навчальні дані повинні бути достатньо якісними для ефективності навчання, а вхідні - для точності роботи.

При певній реалізації даними, що подаються на вхід системи може бути аудіосигнал, що поступає в режимі реального часу, але легше використовувати колекцію готових записів.

1.3 Розгляд задачі класифікації емоцій за мовою

Мовний аудіосигнал містить в собі інформацію різного роду, тож він вважається складним. Мовні сигнали містять інформацію і про самого спікера, і про суть сказаного, і про емоції. Є слова (у всіх мовах), які можуть використовуватись в різному контексті для донесення абсолютно різних емоцій.

Суть поставленої у даній роботі задачі можна визначити як виявлення ознак емоцій людини за її голосовим сигналом. Розпізнавання емоційного стану спікера може бути використане задля вилучення певної семантики з мовлення. А це, насамперед, може підвищити ефективність систем розпізнавання мови.

Існує деякий набір загальноприйнятих емоцій. І постає задача натренувати інтелектуальну систему для ідентифікації цих емоцій.

Якщо розглядати задачу глибше, то мова – це певна послідовність звуків, а звук – це суперпозиція звукових хвиль різних частот. Ці звукові хвилі можна характеризувати за амплітудою і частотою (дивіться рисунок 1.2). Щоб записати звуковий сигнал в електронному вигляді, його спочатку ділять на множину проміжків, а потім відбирають певне усереднене значення на кожному з цих проміжків. Це робиться для перетворення механічних коливань на набір чисел, який вже можна використовувати для обробки комп’ютерними системами.

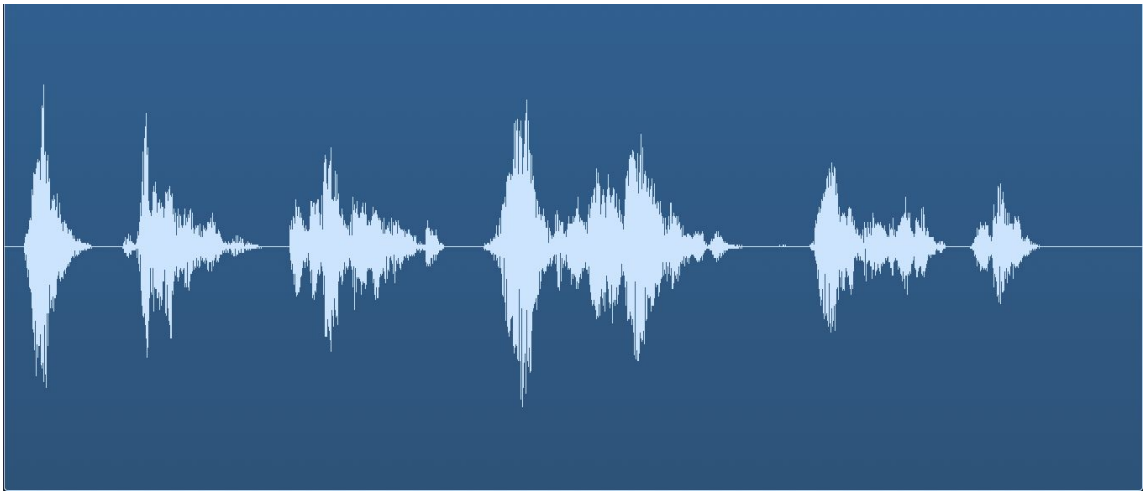

Рисунок 1.2 – Звукова хвиля

Отже задача розпізнавання аудіосигналів (та, зокрема, мовлення) може бути зведена до звичайного зіставлення числової множини цифрового аудіосигналу зі значеннями, що певним чином характеризують емоції [3].

При роботі зі звуковими сигналами, прийнято розбивати вхідні дані на невеличкі числові проміжки - фрейми. Ці фрейми повинні попарно перетинатися. Такий підхід дозволяє забезпечити стійкість методу розпізнавання до можливої різкої зміни гучності мовлення, швидкості або тембру голосу спікера.

Фрейми використовуються в роботі з аудіосигналом, оскільки аналіз хвиль на певному проміжку є більш зручним та ефективним, аніж аналіз в конкретних моментах. Перетинання фреймів допомагає зробити результати аналізу більш згладженими, оскільки ми начебто використовуємо невелике вікно, що рухається вздовж звукового сигналу.

В ході роботи насамперед з кожного запису вилучаються ознаки аудіосигналу. Це робиться задля виявлення того, як пов’язані між собою емоції та мовні шаблони. Вилучені на цьому кроці ознаки є основою навчання і подальшого тестування розпізнавання емоцій. Як вже було сказано вище, саме розпізнавання емоцій проводиться за допомогою певного класифікатора.

Отже вибір ознак, які будуть використовуватись за розпізнавання емоцій, є доволі важливою частиною розв’язання поставленої задачі.

Загалом, типи ознак діляться на два типи:

Першим типом є просодичні ознаки. До них, головним чином, належать: гучність мови, енергія та висота тону, фундаментальна частота,.

Другий тип - це пектральні ознаки. Серед них виділяють:

- лінійні прогнозовані кепстральні коефіцієнти (англ. LPCC - Linear predictive cepstrum coefficients);
- Барк-частотні кепстральні коефіцієнти (англ. BFCC - Bark-Frequency Cepstral Coefficient);
- мел-частотні кепстральні коефіцієнти (англ. MFCC - Mel-frequency cepstral coefficients).

Найчастіше в реалізаціях розпізнавання емоцій за аудіосигналом використовується MFCC, оскільки серед всіх експериментів він показує найвищу ефективність [20].

Найчастіше, в роботах з розпізнавання емоцій за мовою використовується одноетапний підхід. Він полягає в таких діях: спочатку підраховуються акустичні ознаки для всього висловлювання, а потім за ними навчається якась класифікаційна модель. В якості класифікатора найчастіше виступають глибокі нейронні мережі [9][10] та метод опорних векторів [11][12].

В останні роки стає все більш популярною нестандартна постановка задачі. Вважається, що будь-яку емоцію можна описати за допомогою двох або трьох параметрів. За кордоном часто використовують VAD-модель [13], в якій наявні такі метрики, що змінюються протягом мовлення:

- valence (позитивна чи негативна);
- arousal (несподіваность);
- dominance (контрольованість).

Вважається, що в цьому просторі мовлення попадає в області, які належать конкретним емоціям. Головною перевагою такого метода є в більшій інформативності, в порівнянні з дискретними класами. А недоліком є суб'єктивність методу і те, що базу даних насправді доволі важно коректно розмітити.

1.4 Вилучення ознак з аудіосигналу

Дослідження показують, що потенційно найефективнішими параметрами для розпізнавання емоцій є такі параметри як: MFCC, фундаментальна частота (F0), LPCC та енергія.

Розглянемо деякі групи показників, які використовуються в задачах розпізнавання емоцій мовлення найчастіше.

1.4.1 Мел-частотні кепстральні коефіцієнти (MFCC)

Мел-частотні кепстральні коефіцієнти як ознаки розпізнавання аудіосигналів, вживаються найчастіше. Ефективність розпізнавання за використання MFCC є достатньо високою, щоб дослідники раз за разом використовували саме цей параметр.

Загалом MFCC це своєрідне представлення енергії спектру сигналу. Мел-кепстральних коефіцієнти мають підвищену стійкість до перешкод та дозволяють приймати достовірні рішення на відносно коротких інтервалах

аналізу мови. Основною ідеєю методу Мел-кепстральних коефіцієнтів є максимальне наближення інформації до тієї, що надходить на слуховий аналізатор мозку людини. Ознаки, побудовані на основі Мел-кепстральних коефіцієнтів, враховують психоакустичні принципи сприйняття мови, оскільки використовують мел-шкалу, пов'язану з критичними слуховими смугами (Рис. 1.3).

Рисунок 1.3 - Загальний алгоритм отримання кепстральних коефіцієнтів

Отже, розглянемо, на чому побудовані дані ознаки.

Мел - це одиниця висоти звуку, яка базується на сприйнятті цього звуку органами слуху людини або, іншими словами, своєрідне уявлення енергії спектра сигналу, яке зазвичай є вектором з тринадцяти дійсних чисел.

Кепстр (cepstrum), в свою чергу, це результат дискретного косинусного перетворення від логарифма амплітудного спектра сигналу [4].

Перевагами MFCC є, насамперед те, що використовується спектр сигналу. Це дозволяє в аналізі враховувати те, що сигнал, який подається, має в основі хвильову природу.

Також спектр проектують на одноіменну мел-шкалу, що дозволяє виділяти найбільш сприйнятні людиною звукові частоти (Рис. 1.4).

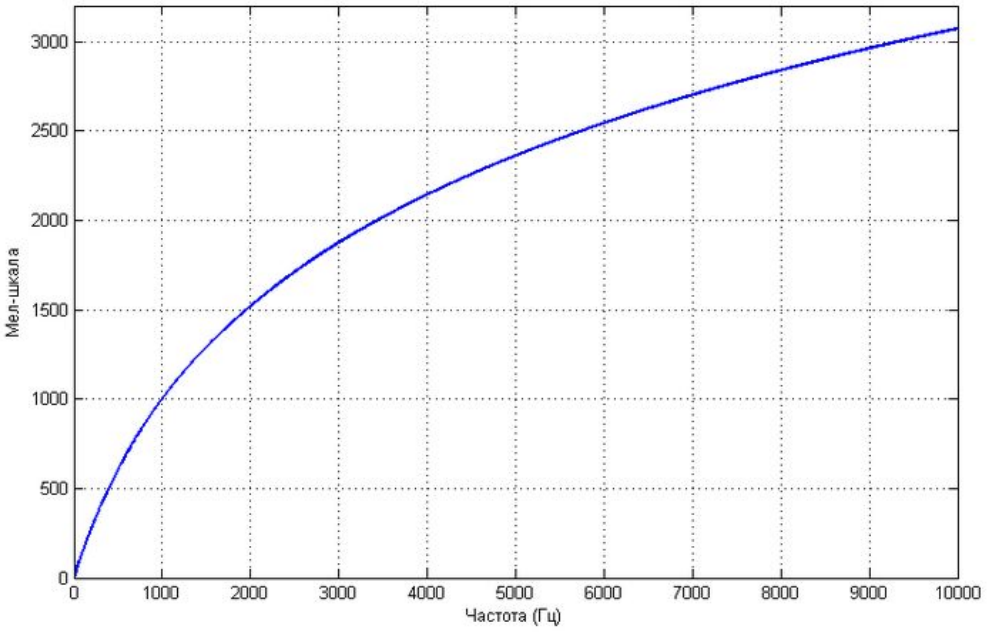

Рисунок 1.4 – Загальний вигляд мел-шкали

Дослідник може обмежити кількість обчислюваних коефіцієнтів будь–яким числом. Це дозволяє стискати фрейм, що в свою чергу дозволяє зменшити кількість інформації для подальшої обробки, а це має колосальне значення в задачах роботи з великими масивами даних.

Кепстральні коефіцієнти є надійними параметрами в розпізнаванні мови. Їх отримують із коефіцієнтів перетворення Фур'є логарифма величини спектру сигналу.

Тож, кепстр сигналу $x(n)$ можна визначити за допомогою перетворення Фур'є:

$$CC(n) = FT^{-1}\{log(FT\{x(n)\})\} \quad (1.1)$$

Частота, що міститься в аудіосигналі не може відповідати лінійній шкалі. Для цього вводиться мел-шкала. Зв'язок між мел-частотою та частотою сигналу можна виразити формулою:

$$F_{mel} = 3233 * log_{10}\left(1 + \frac{F_{hz}}{1000}\right) \quad (1.2)$$

Як видно, обернене перетворення Фур’є було замінено косинусоїдальним перетворенням. Це дозволяє зробити обчислення простішими.

До речі, принцип роботи барк-частотних кепстральних коефіцієнтів загалом такий самий, як у мел-частотних (див. Рисунок 1.3). Тільки для деформації частотної шкали використовується не Мел-шкала, а Барк-шкала. Шкала барків пов'язана з критичними смугами слуху. Оскільки ширина цих смуг (в герцах) нерівномірна, збільшується зі зростанням частоти звукових коливань, то шкала також є нерівномірною.

1.4.2 Лінійні прогнозовані кепстральні коефіцієнти (LPCC)

Лінійні прогнозовані кепстральні коефіцієнти (LPCC) містять ознаки певного каналу мови. Якщо розглянути одну людину у розрізі різних емоційних станів, то вона матиме також різні характеристики каналу. Таким чином ми маємо можливість виявити ознаки для розпізнавання різних емоцій, що проявляються в мовленні.

Голосовий трактр мовного сигналу представляється як фільтр з коефіцієнтами, що постійно оновлюються. Він порушується кожні 15-30мс деяким періодичним чи шумоподібним сигналом. Голосовий кодер вводить певний синтез фільтра, і тим самим моделює голосовий тракт мовного аудіосигналу, тобто таким чином може встановити період та тип збудження

(чи це шум, чи послідовність імпульсів). В якості критерію синтезу фільтра, зазвичай, використовуєть мінімізацію середньоквадратичної похибки, яка вираховується таким чином, що береться різницю фрагментів вхідного мовного сигналу і фрагментів, синтезованих кодером із заданими коефіцієнтами та обчислюється зважеа сума їх квадратів [6].

1.4.3 Енергія та висота тону

Енергія - важливий параметр в мовному сигналі. Різні емоції мають різну енергію. Для отримання значення енергії на кожному вікні, використовують short-term функцію.

На всьому сигналі розглядаються:

- середнє значення енергії;
- найбільше її значення;
- дисперсія енергії та її діапазон;
- контур енергії.

Висота аудіосигналу містить дані про емоції, оскільки вона залежить від того, наскільки напружені голосові зв’язки та яке значення має субгортальний тиск повітря. Тож наступні показники висоти будуть відрізнятися в різних емоційних станах: середнє значення, дисперсія, діапазони дисперсій.

Фундаментальна частота (F0) - це швидкість вібрації голосу.

На основі висоти сигналу також вираховуються середнє значення, медіана, відхилення, максимум та мінімум для вектора та похідної від нього.

1.5 Огляд сучасних методів класифікації

Класифікатором називають систему розподілення деяких даних по класам за наперед визначеними ознаками.

Класифікація є одним із розділів машинного навчання і вона використовується для вирішення поставленого завдання.

Загальний принцип класифікації такий. Маємо деякий пул об’єктів, що можуть бути розділені певним чином на класи. Є датасет, який являє собою кінцеву множину об’єктів, класи яких заздалегідь відомі. Цей датасет називаєть ще навчальною вибіркою. Також є вибірка, в якій приналежність об’єктів до класів не є відомою. Задача - побудувати алгоритм, який зможе вірно класифікувати будь-який об’єкт множин.

Постановку задачі класифікації можна подати у такому вигляді:

X – множина описів ознак, які мають об’єкти, а Y – деяка (скінчена) множина класів. Маємо невідому цільову залежність, певне відображення $y^* : X \to Y$. Значення цієї залежності відомі лише за об’єктами навчальної вибірки $X^m = \{(x_1, y_1), \ldots, (x_m, x_m)\}$. Потрібно побудувати алгоритм, який буде здатним класифікувати будь-який об’єкт з множини X.

Є також ознаки, які являють собою деяке відображення:

$$f : X \to D_f, \tag{1.3}$$

де D_f – це множина допустимих значень певної ознаки.

Якщо ми маємо задані ознаки $f_1, \ldots, f_n$, то описом ознак об’євекту буде вектор $x = (f_1(x), \ldots, f_n(x))$, $x \in X$.

Без втрати загальності можна вважати описи ознак і самі об’єкти - тотожніми поняттями. А множина $X = D_{f_1} \times \ldots \times D_{f_n}$ називається простором ознак [5].

Ознаки можна умовно поділити на такі типи (в залежності від типу множини D_f):

- бінарні ознаки (якщо $D_f = \{0, 1\}$);
- номінальні ознаки (якщо D_f є скінченою множиною);
- порядкові ознаки (якщо D_f є скінченою впорядкованою множиною);
- кількісні ознаки (якщо D_f є множиною дійсних чисел).

В даній роботі розглядається номанільна множина ознак.

Важливою частиною розпізнавання емоцій мовлення є, власне, класифікація.

Існує багато різних класифікаторів, як-то: прихована марковська модель, гаусівська модель суміші, машина опорних векторів, штучні імунні системи.

Також можуть використовуватись нейронні мережі, наприклад згорткова нейронна мережа, рекурентна нейронна мережа, багатошаровий перцептрон та обмежена машина Больцмана. Розглянемо детальніше деякі з наведених класифікаторів.

1.5.1 Нейронні мережі

Згорткові нейронні мережі (ЗНМ) мають змогу виявляти ознаки вищих рівнів та є інваріантними до місцевих спектральних та не постійних зрушень.

Тоді як рекурентні нейронні мережі (РНМ) вважаються досить ефективними інструментами для вивчення довготривалого часового контексту в звукових сигналах.

В даній роботі використовуються обидва підходи, побувані у конструкцію, яку називають згортковою рекурентною нейронною мережею (ЗРНМ). Вона комбінує в собі переваги обидвох видів мереж та може ефективно застосовуватись для вирішення задачі класифікації емоцій. Наведений метод є певною модифікацією, точніше тимчасовою операцією згортки (задля отримання оцінки всього файлу, замість звичайної оцінки на рівні фрейму) та спеціально налаштовуваного гіперпараметру для викликів, які використовуються у РНМ.

Даний підхід, що являє собою поєднання ЗНМ і РНМ, було досить ефективно використано в реалізації задачі класифікації музики [6].

Нейронні мережі часто використовують для розпізнавання, оскільки вони здатні виявити навіть нелінійні межі для розподілення лінійних станів.

Загальний вигляд нейронних мереж зображено на рисунку 1.5.

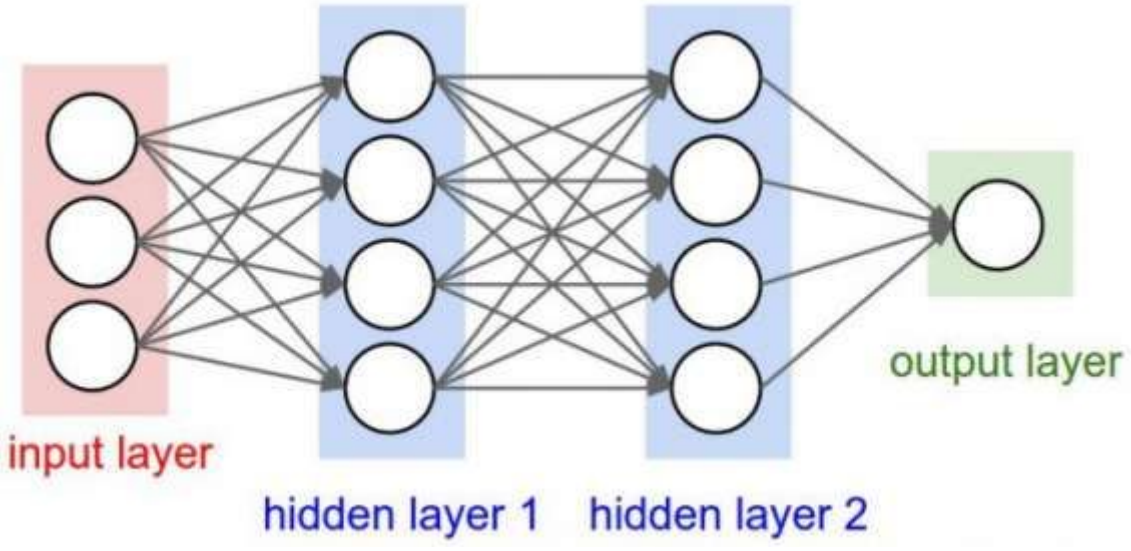

Рисунок 1.5 – Зображення архітектури нейронної мережі

На практиці штучні нейронні мережі можуть сягати близько 70% точності у розпізнванні.

Багатошаровий перцептрон досяг точності рівня 68.10%. при розпізнавання без дикторонезалежності.

Схожий результат було отримано при використанні методу глибокого навчання обмеженої машини Больцмана. Точність сягала 69.14%.

Згорткові нейронні мережі дають приблизно 71% точності для систем розпізнавання з дикторозалежністю [7][8].

1.5.2 Прихована марковська модель

Прихована марковська модель — є статистичною марковською моделлю. У ній у система, яка піддається моделюванню, розглядається у вигляді марковського процесу із станами без спостереження. Загалом, приховану марковську модель можна представити у вигляді найпростішої динамічної баєсової мережі (Рис. 1.6).

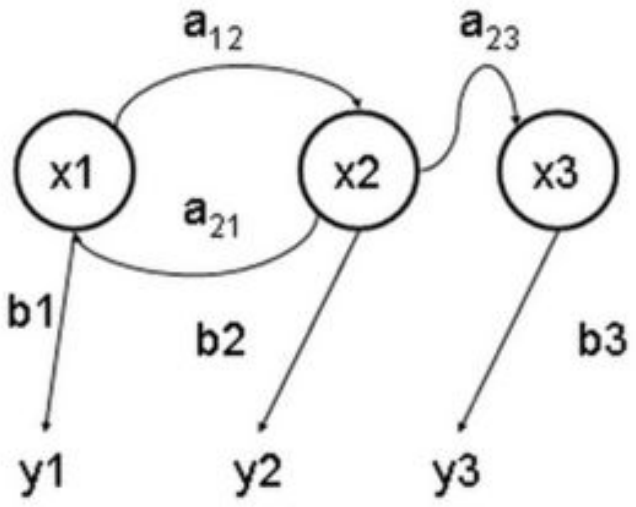

Рисунок 1.6 – Схематичне зображення прихованої марковської моделі

Переваги даної моделі в тому, що тимчасову динаміку характеристик мови можна отримати через матриці переходів.

Суть методу в тому, що є невідомий набір прихованих станів $(x_1, x_2, ..., x_n)$, виявлення яких в певній послідовності (обумовленою ймовірностями $(a_1, a_2, ..., a_n)$) з деякими ймовірностями $(b_1, b_2, ..., b_n)$ приведе до набору спостережуваних результатів $(y_1, y_2, ..., y_n)$.

Для вирішення задачі розпізнавання емоцій мови з використанням прихованих марковських моделей, насамперед датасет сортують відповідно до обраного режиму класифікації, після цього вилучають всі обрані ознаки з аудіосигналу. Вилучені ознаки додають до бази даних. Потім, відповідно до режимів, які генерують випадкову послідовність станів та виходів з моделі, сформовуються матриці переходів та виходів. І наприкінці розраховується оцінка ймовірності послідовності станів з використанням одного з алгоритмів навчання ПММ (в даній роботі розглядається алгоритм Вітербі).

У задачах розпізнавання мови, приховані марковські моделі ефективно моделюють послідовності, які містять в собі декілька слів, Також дані моделі

майже не мають залежності від темпу мовлення. І однією з суттєвих переваг ПММ є досить велика швидкодія.

Приховані марковсткі мережі навчаються для всіх обраних емоцій окремо, а потім тестовий зразок класифікується системою відповідно до моделі, що буде ілюструвати похідну послідовність ознак найкращим чином. В розпізнаванні емоцій даний метод сягає точності у 76.12%, при використанні спектральних ознак.

Недоліком прихованих марковських моделей є, насамперед, те, що ознаки, що виявляються з сигналу, повинні не тільки містити в собі інформацію щодо емоцій, але такой відповідати заявленій структурі моделі.

1.5.3 Штучні імунні системи

Щоб мати можливість застосовувати принципи штучних імунних систем (ШІС) при вирішенні певних задач, необхідно використовувати алгоритми (методи) ШІС.

Існують три основні алгоритми, що постійно використовуються дослідниками в розробці класифікуючих систем, а саме: Алгоритм негативного вибору, Алгоритм клонального відбору та Алгоритм імунної мережі.

Алгоритм негативного відбору (див. рис. 1.7) базується на механізмі негативного відбору, успадкованому в імунній системі для захисту організму від самореактивних лімфоцитів, де він виявляє невідомі антигени, не реагуючи на рідні клітини. Це одна з найпопулярніших моделей, яка вплинула на розвиток багатьох існуючих штучних імунних систем. Концепція негативного

відбору описує процес дозрівання Т-клітин в імунній системі, де будь-яка рідна клітина ліквідується, якщо Т-клітина в тимусі розпізнала це. Алгоритм генерує набір детекторів, видаляючи будь-якого кандидата-детектора, який відповідає елементам, з групи самостійних зразків.

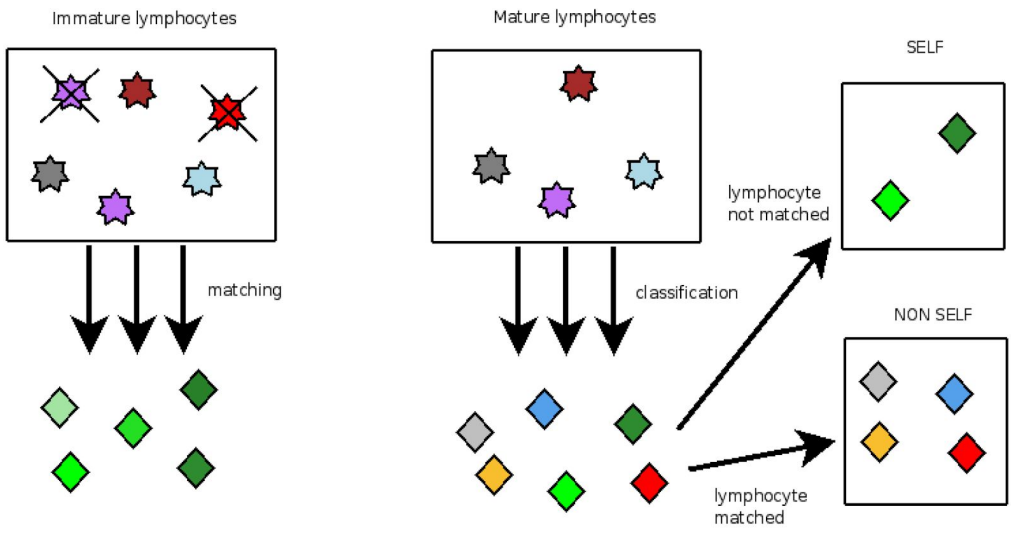

Рисунок 1.7 - Модель негативного відбору

Ідея алгоритму клонального відбору базується на теорії клонального відбору Бернета. І на відміну від алгоритму негативного відбору алгоритм клонального відбору працює як на В, так і на Т-клітинах.

Алгоритм вмонної мережі базується на теорії, що імунна система складається з регульованої мережі молекул і клітин, здатних розпізнавати один одного, навіть якщо антигенів не було представлено. В-клітини пов'язані між собою, зокрема, способами стабілізації мережі, коли кожні дві В-комірки повинні перевищувати певний поріг спорідненості, а сила їх зв'язку повинна бути прямо пропорційною цьому рівню спорідненості.

Способи подання дозволяють створювати абстрактні моделі імунних органів, клітин або молекул; механізми оцінювання, які називають також функціями афінності, дозволяють кількісно оцінити взаємодії цих «штучних

імунних органів», а процедури адаптації, виражені у вигляді безлічі загальних алгоритмів досягнення мети, управляють динамікою ШІС.

1.5.4 Метод опорних векторів

Метод опорних векторів (МОВ) використовується у машинному навчанні та являється методом аналізу даних для задач класифікації та регресійного аналізу. Базується метод на основі алгоритмів навчання, що звуться опорно-векторними машинами. Принцип дії наступний. Для певного заданого (відміченого до однієї з двох категорій) набору тренувального датасету, алгоритм будує модель, що має завдання віднести ці зразки до певної категорії, працюючи на основі неймовірнісного бінарного лінійного класифікатору.

Якщо коротко, то дана модель являє собою представлення зразків у вигляді точок у просторі. І ці точки відображені таким чином, що зразки з окремих класів можна розділити деякою межею (Рис. 1.8). На меті стоїть знайти найкращу таку межу. Тестові зразки, при класифікації, переносяться на даний простір і система робить передбачення щодо їхної належності до певної категорії, використовуючи інформацію про те, до якого боку межі вони потрапили.

Головна ідея методу опорних векторів полягає в тому, щоб перетворити початковий вхідний набір у багаторозмірний простір параметрів, за допомогою використання ядерної функції (Kernel Function).

В методі опорних векторів точка даних - це багатовимірний вектор. Задача - дізнатися, чи є можливість якимось чином розділити отримані точки

такою ж багатовимірною гіперплощиною. Цей процес називають лінійним класифікатором. Можна побудувати дуже багато гіперплощин, що можуть розділити певні дані. Один із популярних варіантів вибору найкращої гіперплощини є обрання найбільшого проміжку між класами. Тобно обирається така гіперплощина, аби відстань від цієї гіперплощини до найближчих до неї точок з обидвох боків була максимальною. Тож дана гіперплощина, якщо її можна побудувати, називається максимально розділовою гіперплощиною (англ. maximum–margin hyperplane), а лінійний класифікатор, який вона визначає, називається максимально розділовим класифікатором (maximum margin classifier).

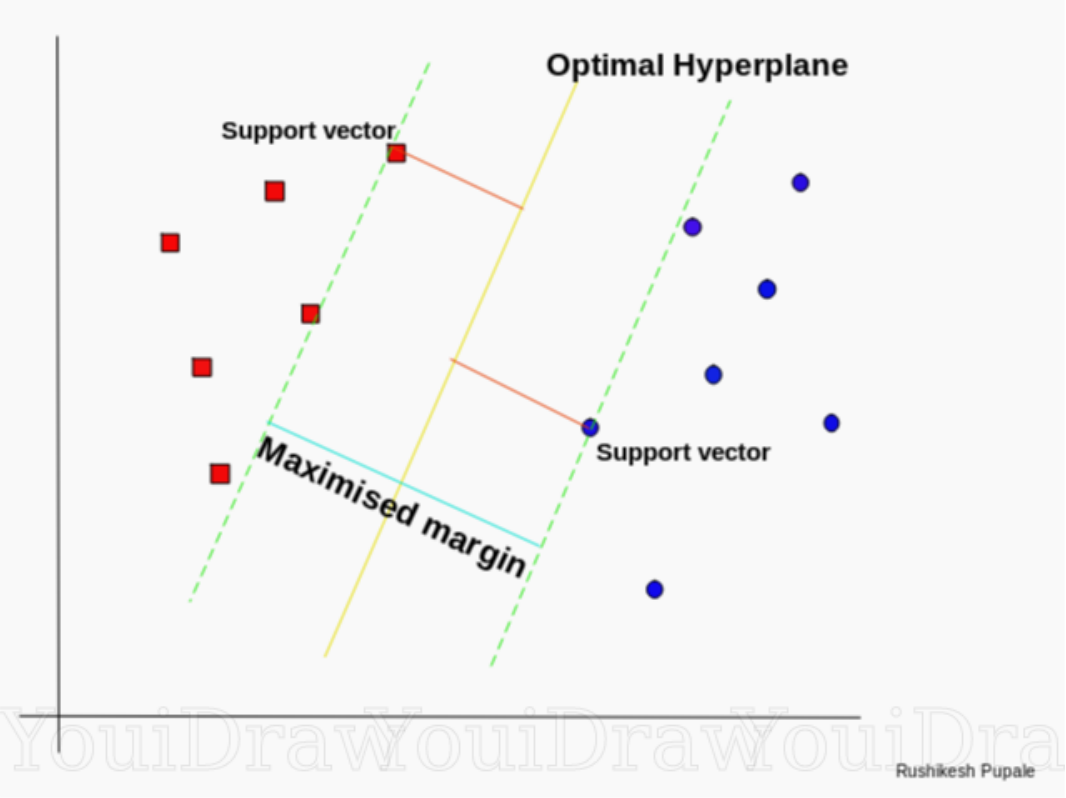

Рисунок 1.8 – Приклад роботи методу опорних векторів

Отже, основна задача даного методу - побудова гіперплощини (або набору гіперплощин) у багатовимірному просторі так, щоб ці площини можна було використовувати для класифікації, регресії та інших задач.

Точність методу опорних векторів в задачах розпізнавання емоцій мовлення сягає близько 80%.

1.6 Огляд реалізованих програм розпізнавання емоцій мови

Загалом, спроби розпізнавати емоції почали набирати популярність лише нещодавно.

Отже, з відомих на сьогодні реалізацій розпізнавання емоцій можна виділити EMOSpeech, EmoVoice та Affectiva.

1.6.1 EMOSpeech

EMOSpeech – є програмою взаємодії робітниців колцентрів з їх клієнтами. Вона була першим програмним забезпеченням, що мало на меті розпізнавання емоцій і таким чином дозволяла центрам для обробки викликів пропускати через аналітичну систему всі дзвінки, що були записані. Це забезпечувало зворотній зв’язок, що працював у реальному часі.

Реалізовано це рішення з огляну на простоту та легувсть використання: зручний інтерфейс та проста інтеграція в існуючих платформах для колцентрів.

Модель базується на визначенні емоцій у тривимірному безперервному просторі. В цьому просторі кожен вимір являється емоційною примітивністю, що проявляється всіма людськими емоціями: валентністю, домінуванням та активацією.

Даний підхід має в собі немалий потенціал для моделювання процесу появи емоцій у нашому житті. У нормальному сценарії емоції не генеруються в незмішаній модальності. Вони зазвичай являють собою складні емоційні стани, які в свою чергу - це суміш різних емоцій, які мають різний ступінь інтенсивності. Використовують метод побудови емоцій, що мають за основу статистичні моделі та нечітку кластеризацію для отримання вичерпної інформації. Також система моделює певний зв’язок між особливостями аудіосигналу та емоційними примітивами, беручи за основу використання статистичного підходу на основі акустичної інформації. Інтерпретація Зв’язка емоційних примітивів зі станами відбувається з використанням нечітких логічних моделей, які, власне, і моделюють емоційні явища.

Компанія протестувала близько семи тисяч акустичних функцій, включаючи просодію, якість передачі голосу та спектральну інформацію.

1.6.2 EmoVoice

EmoVoice являє собою всеохоплюючу систему для розпізнавання емоцій. Може працювати у режимі реального часу та відштовхується лише від акустичних властивостей мовлення (не використовуючи інформацію про слова).

EmoVoice містить два модулі:

- один для незалежного аналізу емоційної бази мови;
- інший - для відстеження онлайн впливу мовлення, під час розмови.

Система являє собою певний набір інструментів, за допомогою яких створюються власні емоції для розпізнавання в реальному часі.

На сьогодні EmoVoice має 2 алгоритми класифікації:

1. Наївний Класифікатор Байєса (NB).
2. Метод Опорних Векторів (SVM).

Наївний Класифікатор Байєса вважається достатньо швидким, і у випадку багатомірних векторів параметрів також. Тож він ідеально підходить для аналізу даних в режимі реального часу.

Однак, класифікаційні показники трохи нижчі за метод опорних векторів.

Вибір функції, у поєднанні зі зменшенням кількості ознак до меньше ніж ста, метод опорних векторів так само проявляє велику ефективність у розпізнаванні в режимі реального часу.

1.6.3 Affectiva

Affectiva - це компанія, що спеціалізується на технологіях розпізнавання емоцій. Вона розробила програмне забезпечення розпізнавання людських емоцій на основі сигналів обличчя, фізичних та психологічних реакцій людини. Серед комерційних додатків ця технологія використовується, щоб допомогти брендам вдосконалити свої рекламні та маркетингові повідомлення.

Технологі дозволяє програмним додаткам використовувати веб-камеру для слідкуванням за посмішками, насупленністю та зморшками користувача, що дозволяє зрозуміти ступінь здивованості, радості чи розгубленості користувача.

Технологія також дозволяє виміряти частоту серцевих скорочень людини за допомогою веб-камери без використання датчика. Це досягається шляхом відслідковування змін кольора обличчя людини, що відбувається кожен раз, коли б’ється серце.

Ігри використовують цю технологію, щоб адаптовуватись під рівень напруженості людини за його емоційним станом.

Частина продукту, яка відповідає за роботу з мовою людини, аналізує не те, що людина говорить, а те - як саме, спостерігаючи за змінами в паралігвістиці мови, гучності та темпі голосу, щоб розрізнити мовніподії, емоції та стать. Підхід з низькою затримкою є ключом до можливості розробки додатків розпізнавання емоцій в режимі реального часу.

Їх перший мовний продукт аналізує лише попередньо записаний сегмент формату MP3. Вихідний файл містить аналіз мовних подій, що відбуваються на записі. Продук, що буде аналізувати мову в режимі реального часу (Emotion SDK) поки що недоступний.

Компанія використовує методи машинного навчання у своїй роботі, постійно поповнює свою базу даних. Точніть розпізнавання досягає приблизно 90%. Тестові дані доволі об’ємні, багаті етнічною та віковою різноманітністю, мають різний рівень зашумленості.

Великий плюс стратегії компації в тому, що вони комбінують аналіз аудіо та відео, для більш точного відображення реального емоційного стану людини.

1.7 Висновки до розділу

У розділі було розглянуто певні наявні методи класифікації та їх реалізації. До того ж було розглянуто різноманітні пули ознак, які мають в собі інформацію щодо емоційного стану спікера та можуть використовуватись у задачах з класифікації.

В межах поставленої задачі, можна припустити, що найефективнішими є методи, які можуть гарно працювати із великою кількістю незалежних одні від одних даних.

Висувається гіпотеза, що найефективнішими рішеннями при розв’язанні задачі розпізнавання емоцій по голосу є

- прихована марковська модель;
- метод опорних векторів;
- штучні нейронні мережі (зокрема згорткова рекурентна нейронна мережа).

Ознак найкраще виявляти, використовуючи мел-частотні кепстральні коефіцієнти, енергію та висоту сигналу. В припущенні, це може дозволити збільшити точність в розпізнаванні емоцій в мові людини.

РОЗДІЛ 2. МАТЕМАТИЧНІ ОСНОВИ РОЗПІЗНАВАННЯ ЕМОЦІЙ ЛЮДИНИ ЗА МОВЕННЯМ

2.1 Постановка задачі розпізнавання емоцій

Щоб розробити інтелектуальну систему розпізнавання емоцій людей за мовою, потрібно в першу чергу вирішити, які конкретно ознаки використовуватимуться в дослідженні. Точність розпізнавання напряму залежить від підібраних ознак.

Найпопулярніші ознаки ми розглянули в РОЗДІЛІ 1.

Порівнюючи розглянуті у першому розділі існуючі реалізації задачі класифікації мови за емоціями та ознаки, які в цих реалізаціях використовувались, було зроблено висновок, що оптимальним набором ознак є висота звуку, енергетичність та мел-кепстральні коефіцієнти (див. Рис. 2.1), які будуть доставатися з наборів фреймів.

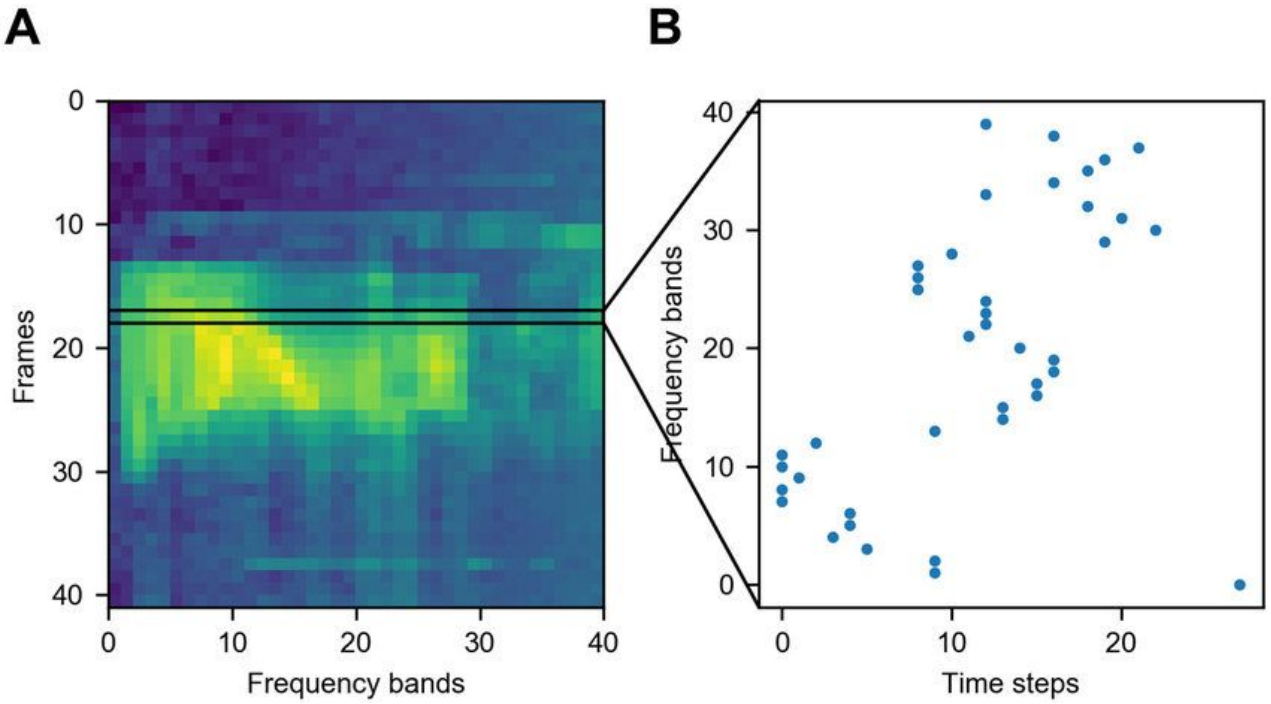

Рисунок 2.1 - MFSC спектрограма

2.2 Датасет та попередня обробка даних

У даній роботі використовується база даних RAVDESS (The Ryerson Audio-Visual Database of Emotional Speech and Song). Це база голосових даних 24 актори та акторки, які записали по декілька прикладів (фрази “Kids are talking by the door” та “Dogs are sitting by the door”) на кожну з восьми наступних емоцій:

- Нейтральність (Neutral);
- Спокій (Calm);
- Радість (Happy);
- Сум (Sad);
- Злість (Angry);

- Страх (Fearful);
- Відраза (Disgust);
- Здивованість (Surprised).

Розбиття датасету по емоціям див. на Таб. 2.1

Таблиця 2.1 - Розподіл емоцій в датасеті RAVDESS

Emotion	Speech Sample Count	Song Sample Count	Summed Count
Neutral	96	92	188
Calm	192	184	376
Happy	192	184	376
Sad	192	184	376
Angry	192	184	376
Fearful	192	184	376
Disgust	192	0	192
Surprised	192	0	192
Total	1440	1012	2452

Отже, загалом, датасет містить 2452 файли. У записі та оцінці RAVDESS брало участь більш ніж 250 осіб. Серед файлів є аудіо-текст, аудіо-пісні та відео. В рамках даної роботи розглядається лише аудіо-текст.

В рамках роботи вирішено з’єднати такі емоції як нейтральна та спокій, оскільки вони по суті досить схожі і в даному дослідженні нема сенсу перевантажувати систему завеликою кількістю класів.

Насамперед, перше, що потрібно зробити зі звуковими даними - очистити їх від шумів. Попередня обробка конче необхідна, коли не використовується стандартна БД. Метою попередньої обробки є підвищення високих частот сигналу і отримання плоского частотного спектру сигналів та частотних характеристик.

Щоб отримати відповідну амплітуду на кожному фреймі, потрібно, щоб в записах переважали високі частоти. Зазвичай, цей процес проходить за допомогою фільтрування аудіосигналу з використанням фільтра із скінченими імпульсними характеристиками першого порядку. Передаточна функція такого фільтра в z-перетворенні виглядає так:

$$H(z) = 1 - \alpha * z^{-1}, \tag{2.1}$$

де α – це параметр, отриманий на попередніх кроках.

Для роботи з сигналом, потрібно його розбити на фрейми. Тож поділимо всі аудіосигнали (в даній роботі розглядаються сигнали з розширенням .wav) на певні інтервали. В роботі [15] показано, що в інтервалах довжиною 25 мс і більше міститься достатньо інформації щодо емоційного стану. Така довжина відповідає одній чи декільком фонемам. Тож в цій роботі довжина фреймів буде дорівнювати теж 25 мс. А довжина перетину фреймів - 50%.

Розглянемо більш детально алгоритм вилучення MFCC. Спершу за допомогою ДПФ рахується спектр одного фрейму:

$$X(n) = \sum_{k=0}^{N-1} x(k) e^{-2\pi i n \frac{k}{N}}, \ 0 \le n < N \tag{2.2}$$

Розрахувавши ДПФ для всіх фреймів, отримаємо спектр сигналу (див. Рис. 2.2)

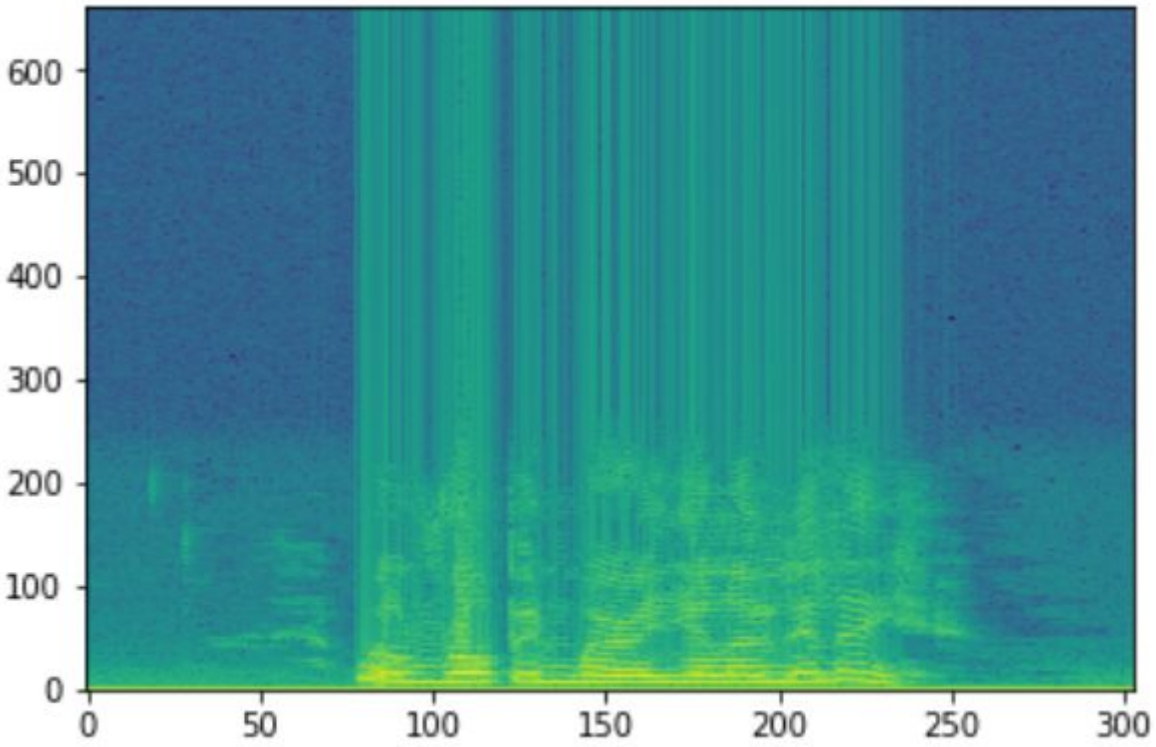

Рисунок 2.2 - Спектр вхідного сигналу

До результату застосовуємо віконну функцію Хемінга. Вона допомагає трохи згладшувати значення на краях фреймів:

$$H(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{(N-1)}\right) \quad (2.3)$$

Тоді в результаті матимемо вектор:

$$X(n) = X(n) * H(n), \quad 0 \leq n < N \tag{2.4}$$

Отримали частоту. Треба перетворити її в мел-частоту за допомогою даної формули:

$$M = 1127 * log(1 + \frac{F}{700}) \tag{2.5}$$

Обернене перетворення виглядатиме так:

$$F = 700 * (e^{\frac{M}{1127}} - 1) \tag{2.6}$$

В рамках даної роботи достатньо покласти M = 16 [16].

За допомогою спектрального аналізу можна виявити ознаки мовленневих сигналів, які зумовлені більш за все голосовим трактом (і в часності його формою). Мовні спектральні ознаки, як правило, відбираються як вихідне значення фільтрів, що певним чином проводять інтеграцію спектру у діапазони частот. Зазвичай йдеться про вибірку з двадцяти чотирьох фільтрів-смуг (які в обробці діють схожим чином із обробкою вух людей). Існує численна кількість розроблених таких фільтрів.

Щоб можна було розкладати розглянутий вище спектр за мел--шкалою, треба перетнути фільтри. Кожний фільтр являє собою деяку віконну функцію, що допомагає сумувати кількісні показники енергії на певних діапазонах частот. Так отримуються, власне, мел-коефіцієнти. Якщо знати реальну кількісну характеристику мел-коефіцієнтів та діапазон частот, щодо якого проводиться аналіз, то можна відтворити пул фільтрів (див. Рис. 2.3).

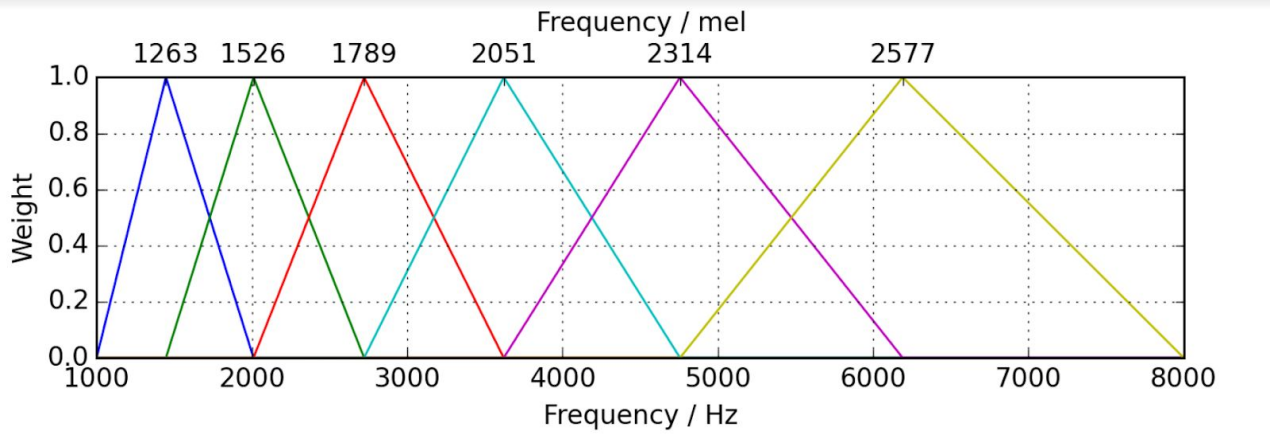

Рисунок 2.3 - Схематичне зображення набору фільтрів

З рисунка видно, що основа фільтру буде ширшою при більшому порядковому номері мел-коефіцієнта.

Формула для підрахування частоти кепстру буде виглядати так:

$$b_m = b_{m-1} + \Delta_m, \tag{2.7}$$

де Δ_m треба обрати таким, щоб отримати десять фільтрів, що будуть розподілені рівномірно.

Наостанок, потрібно розрахувати самі мел-кепстральні коефіцієнти:

$$y_t^{(m)}(n) = \sum_{m-1}^{M} log \left\{ \left| Y_t(m) \right| \right\} * cos \left(\frac{n(m-\frac{1}{2})\pi}{m} \right), \; n = 0, \ldots, L \tag{2.8}$$

Отримана інформація записується у таблицю, яку подають на певний класифікатор. Це будуть вхідні параметри для класифікатора (навчальні дані та тестові).

2.3 Розробка алгоритму класифікації

Майже всі з описаних в даній роботі ознак чутливі до довжини вхідного аудіосигналу. Класифікатори навчаються на ознаках, кожна з яких рахується для усього висловлювання. Але не зовсім коректно вважати, що емоційна складова фрейму повністю співпадає з емоцією усього висловлювання. На жаль, на разі дуже важко виявити, як положення фрейму у висловлюванні впливає на його емоційний стан, оскільки цей вплив буде різнитися в залежності від самої емоції, довжини висловлювання та особливості мови того, хто говорить.

Пропонується метод, який дозволить обійти цей недолік. Класифікація розбивається на 2 етапи (див. рис. 2.4). На першому етапі, як і в розглянутих алгоритмах, висловлючання розбивається на фрейми, які частково перетинаються і для кожного фрейму вираховуються акустичні ознаки. Потім за допомогою навченної нейронної мережі ми отримуємо ймовірність кожного фрейму попасти в одну із заданих емоцій. Для конкретного інтервалу це означає деяке проектування вектора ознак на певний простір, розмірність якого співпадає з кількістю емоцій, що розглядаються. Таким чином, ми отримуємо часовий ряд, що складається з ймовірностей і має час дискретизації такий самий, як довжина фрейму. На другому етапі класифікатор використовує цей часовий ряд, щоб прийняти рішення про все висловлювання.

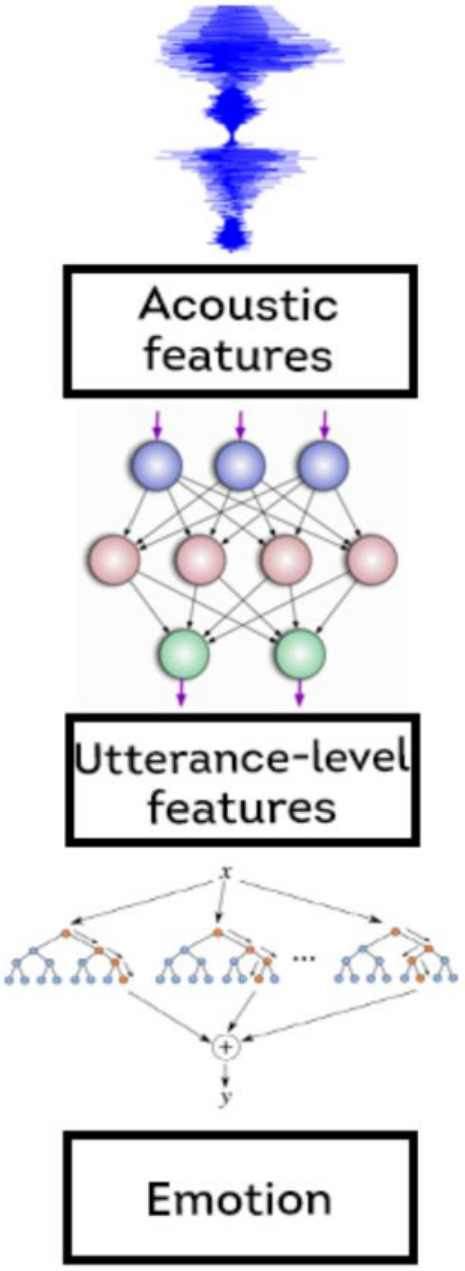

Рисунок 2.4 - Схема двоетапного алгоритму класифікації

Кандидатами на класифікатор для першого етапу були: марковьска прихована модель, згорткова нейронна мережа та рекурентна нейронна мережа.

Марковські приховані моделі досить добре себе показують в задачах роботи із аудіо. Тож для вирішення, який метод використовувати окрім ПММ,

було проведено порівняння останніх двох класифікаторів (див. Таблицю 2.2) з метою обрати найбільш ефективну для поставленої задачі.

Таблиця 2.2 — Порівняння згорткової та рекурентної нейронних мереж

	Згорткова нейронна мережа	Рекурентна нейронна мережа
Вхідні та вихідні дані	Приймає вхід фіксованого розміру та генерує вихід фіксованого розміру;	Може обробляти довільну довжину входу/виходу;
Принцип дії	Використовують схему взаємодії між нейронами. Базуються на схемі організації зорової кори тварин, індивідуальні нейрони яких побудовані таким чином, що вони реагують на області, що накладаються одна на одну, покриваючи все поле зору;	Використовують інформацію щодо часових рядів. Те що відбулося раніше, буде впливати на те, що відбуватиметься далі;
Попередня обробка	Представляє собою варіацію багатошарових перцептронів, які призначені для використання мінімальних об’ємів попередньої обробки;	Може використовувати свою внутрішню пам’ять для обробки довільних послідовностей входів;
Поле використання	Краще використовувати для обробки зображень та відео;	Краще використовувати для аналізу тексту та мовлення.

Оскільки обидві мережі мають свої переваги та недоліки, вирішено використовувати у реалізації усі три моделі і експериментально визначити, яка з них найкраще себе покаже в задачах розпазнавання емоцій в мові.

Замість класичної РНМ, розглянемо її модифікацію під назвою Довга короткочасна пам’ять (long short-term memory - LSTM).

Від класичної версії рекурентних НМ, LSTM відрізняється тим, що містить одноіменні вузли, які використовуються для запам’ятовування значень на короткі та довгі проміжки часу (див. Рис. 2.5).

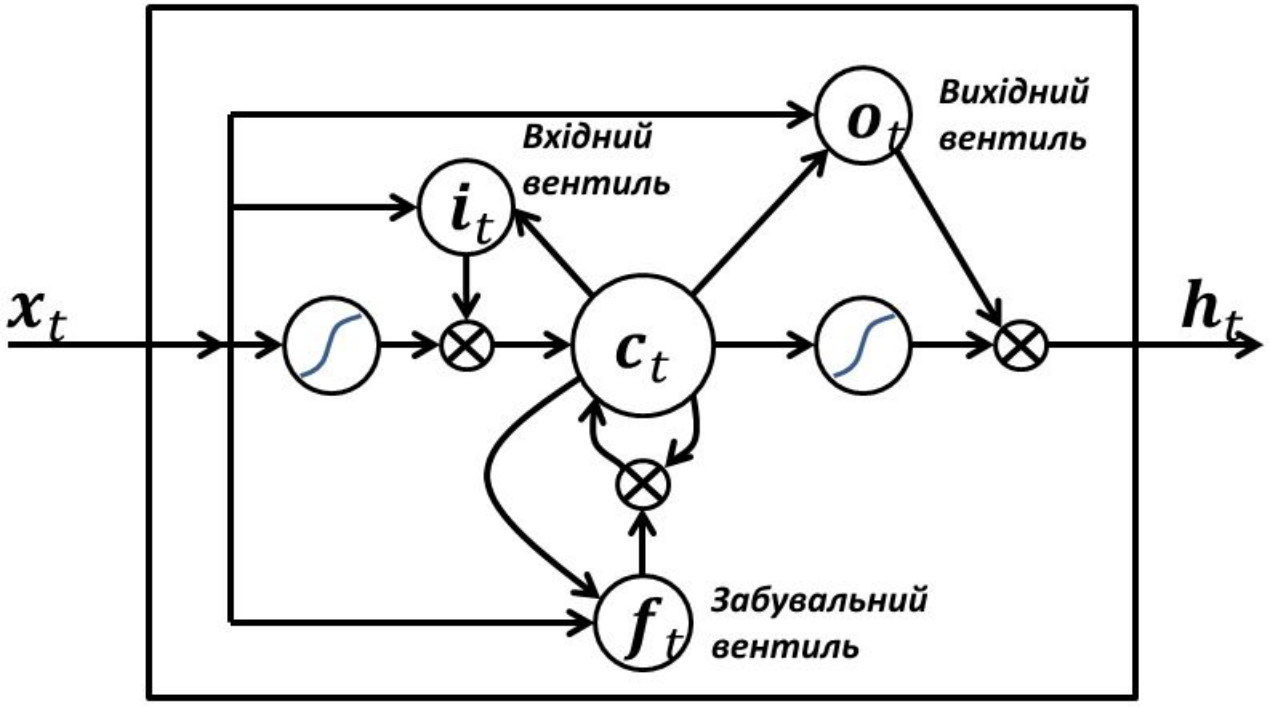

Рисунок 2.5 - Схематичне забраження LSTM

Блоки LSTM, зазвичай, містять 3-4 “вентилі”, що використовуються, щоб керувати зберіганням інформації в пам’яті. «Вхідний вентиль» призначений для керування мірою, до якої нове значення входить до пам'яті. «Забувальний вентиль» призначений для керування мірою, значення якої залишається в пам'яті. І «вихідний вентиль» призначений для керування мірою, до якої значення в пам'яті використовується для обчислення активування виходу блоку.

А вибір класифікатора на другому етапі, зумовлюється його можливостями в навчанні. Оскільки розмір навчальної вибірки на другому

етапі буде співпадати з кількістю висловлювань, а це в рази меньше, аніж кількість фреймів. Тому використання нейронних мереж в цьому випадку себе не окупає. Гарним кандидатом є випадковий ліс (з англ. Random forest). Випадковий ліс - метод машинного навчання, який використовують для вирішення задач з класифікації, регресії. Суть методу в побудові числових дерев прийняття рішень при тренуванні моделі і продукуванні моди для класів (у випадку класифікації) або усередненого прогнозу (у випадку регресії) побудованих дерев. Основним недоліком методу є схильність до перенавчання.

Окрім продемонстрованої гарної точності й стійкості, в порівнянні з іншими методами, він дозволяє порівнювати значимість так званих “ранніх” та “пізніх” ділянок. Забігаючи наперед, виявилось, що в більш пізніх фреймах міститься більш точна інформація щодо емоції висловлювання.

2.4 Висновки за розділом

В цьому розділі ми сформулювали математичну основу завдання розпізнавання емоцій за мовленням.

Було проведено аналіз різноманітних класифікаторів, порівняно їх точність та обрано найефективніший класифікатор.

Також було запропоновано розроблений алгоритм розпізнавання емоцій людини за голосом, який полягає в розбитті процесу класифікації на два етапи:в першому навчання відбувається на фреймах, а другий етап, беручи за

основу результати першого етапу - приймає рішення щодо емоційної наповненості усього висловлювання.

РОЗДІЛ 3. АНАЛІЗ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

3.1 Обгрунтування вибору платформи та мови програмування

Було вирішено реалізовувати програмний продукт на мові програмування Python, оскільки наразі це одна з найбільш пристосованих до машинного навчання мов програмування.

В ході реалізації даної роботи було задіяно такі бібліотеки в мові Python:

- librosa (бібліотека Python для аналізу музики та аудіо);
- numpy (open-source модуль для Python, який надає загальні математичні і числові операції у вигляді попередньо скомпільованих, швидких функцій);
- soundfile (може зчитувати та записувати аудіофайли);
- keras (відкрита нейромережева бібліотека Python, являється надбудовою над фреймворками Deeplearning4j, TensorFlow і Theano; націлена на оперативну роботу з мережами глибинного навчання, спроектована компактною, модульної та розширюєваною);
- sklearn (бібліотека для машинного навчання на Python, за допомогою якої можна реалізувати різні алгоритми класифікації, регресії і кластеризації, в тому числі алгоритми SVM, випадкового лісу, k-найближчих сусідів і DBSCAN, які побудовані на взаємодії бібліотек NumPy і SciPy з Python);
- PyTorch (бібліотека Python, що дозволяє тензорні обчислення з потужним прискоренням GPU).

Більша частина фреймворків мають статичний процес побудови систем. Треба розробити НМ, щоб потім користуватися тією самою структурою

скільки завгодно разів. Мінімальна зміна у способі поведінки нейромережі буде значити, що необхідно розпочинати все з нуля. Серед перерахованих вище бібліотек є ті, що використовуютьб автодиференціацію "Реверс-режим". Ця техніка допомагає міняти нейронну мережу, поки вона працює і робить цез мінімальними втратами.

Також перераховані бібліотеки дозволяють просто вносити та відображати дані, спрощують роботу з ними та допомагають ефективно графічно представляти результати.

Більш детально ознайомитись із причинами вибору саме такої платформи та мови програмування можна у четвертому розділі.

3.2 Аналіз архітектури програмного продукту

Програмний продукт являє собою реалізацію алгоритму розпізнавання емоцій, що використовує перераховані бібліотеки Python для того, щоб будувати класифікатори.

Програма складається з наступних частин:

1. FeaturesExtractor.py – модуль, за допомогою якого ведеться обробка вхідних даних та створюється набір ознак для подальшої роботи з ними. Результатом роботи модуля є файл у форматі .csv.

2. TrainModels.py - модуль, за допомогою якого проводиться двоетапне навчання моделі (в якості класифікаторів першого етапу виступають прихована марковська модель, згорткова нейромережа та

рекурентна нейромережа; на другому етапі використовується випадковий ліс). Навчання відбувається за допомогою графічної карти. Результатом модуля є файл у форматі .h5.

3. CNN.py - реалізована згорткова нейромережа. Архітектурно це 4-шарова ЗНМ з функцією активації ReLU (Rectified Linear Units).

4. RNN.py - реалізована рекурентна згорткова мережа. Було вирішено обрати за архітектурою двошарову РНМ з гіперболічним тангенсом в якості активаційної функції та дропаутом рівним 0.2 (для запобігання перенавченню).

5. HMM.py - реалізована прихована марковська модель. При розробці моделі було вирішено використовувати 20 прихованих станів, оскільки практика показує, зо це найбільш інформативне число [18].

6. MainPart.py - основний модуль. В ньому викликаються всі інші реалізовані модулі та функції.

Реалізований програмний продукт потрібно запускати з консолі (за допомогою команди python або python3), оскільки розробка інтерфейсу не була основною задачею даного дослідження.

3.3 Аналіз результатів роботи програмного продукту

В першу чергу було вирішено протестувати реалізовані моделі в одноетапному алгоритмі, а потім запустити на двоетапному, використовуючи

вже лише той класифікатор, який покаже себе найефективнішим серед інших розглянутих.

Отже, на початку було навчено моделі без використання другого етапу.

Дані було розподілено 80% на навчальну вибірку і 20% на тестовий набір.

Параметри комп'ютера, на якому було реалізовано програмний продукт:

- ОС: Windows 10 Pro;
- Процесор: Intel(R) Core™ i5-8400H CPU;
- Обсяг оперативної пам'яті: 8Гб;
- Відеокарта: NVIDIA GeForce GTX 1050 Ti, 4 Gb.

Реалізований програмний продукт, після проведення навчання всіх згаданих у минулому пункті моделей, надає результати, які описані нижче.

Першою було навчено приховану марковську модель. Експериментально було виявлено, що найоптимальніша кількість станів - 20. Якщо брати менше - гірший результат. Якщо більше - час навчання значно зростає, а ефективність майже не збільшується.

Ознаки, які було виявлено в аудіозаписах було записано в матрицю ознак, яка і використовувалась далі в якості спостережуваних результатів. Навчання ПММ проводилося за допомогою ЕМ-алгоритму, а точніше - за допомогою алгоритму Баума-Велша. Цей агоритм дозволяє виявити ймовірність кінцевої спостережуваної послідовності $y_{t+1},...,y_T$, за умови, що ми почали зі стану i в деякий момент t.

Експериментально було виявлено, що якщо перемішати у випадковому порядку строки матриці ознак, то ефективність алгоритму зростає.

Ефективність алгоритму на семи класах тестової виборки сягла 59.63 %, що є доволі непоганим результатом.

Другою на черзі була рекурентна нейромережа. Було обрано таку архітектуру:

- 2 шари LSTM (512 та 256 нейронів відповідно), після кожного шару використовується активаційна функція - гіперболічний тангенс;
- і ще 2 шари Dense (на 512 та 7 нейронів). Dense - це звичайні лінійні юніти, які просто рахують зважену суму компонентів вхідного вектора.

Після першого шару в якості активаційної функції використовується теж гіперболічний тангенс, після другого - softmax (нормована експоненційна функція):

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.1}$$

Під час навчання використовувалась функція витрат - перехресна ентропія. А в якості оптимізатора - RMSprop (root mean square propagation) - середньоквадратичне розповсюдження.

Точність на тестовій виборці була значно вищою, ніж у ПММ - 64.17%.

Останньою навчалася згорткова нейронна мережа. Її архітектурою послужила вже перевірена мережа - Alexnet. Ця мережа містить п’ять згорткових та три повнозв’язні шари. Вихідні дані теж проходять через функцію втрат softmax (див. рисунок 3.1)

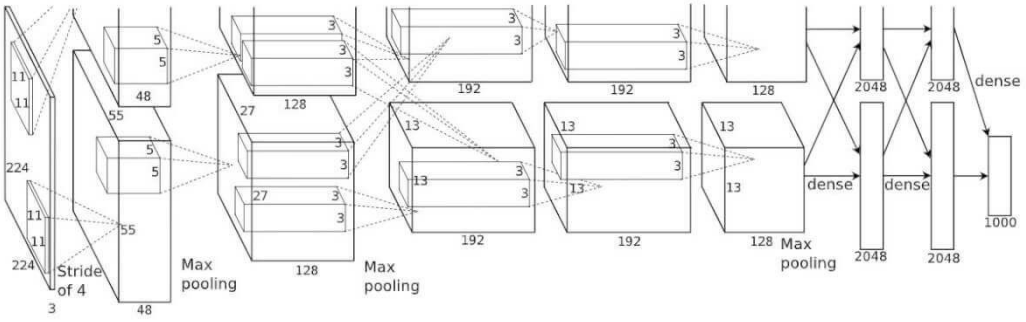

Рисунок 3.1 - Архітектура ЗНМ Alexnet

Згорткова нейромережа дала результат 63.02% на тестових даних. Хоча її архітектура складніша за архітектура рекурентної НМ, точність вийшла нижчою. Отже рекурентна НМ показала себе найкраще в процесі навчання. Це може бути спричинено тим, що ПММ ефективніше застосовується у виділенні фонем та розпізнаванні мови, а ЗНМ доцільніше використовувати для обробки зображень, а не аудіофайлів. З розглянутих класифікаторів, РНМ найкраще справляється з часовими рядами, а доданий шар LSTM за рахунок вибіркової пам’яті, підвищує ефективність класифікації.

Отже, вирішено в двоетапному алгоритмі використовувати саме рекурентну нейромережу. Як вже було сказано, суть двоетапного алгоритму в тому, що до часового ряду, який отримується з класифікації нейронною мережею, застосовується випадковий ліс для прийняття рішення щодо усього висловлювання на основі його фреймів.

Загальна точність моделі з двоетапним алгоритмом є приблизно 68% (див. рис. 3.2). Тобто введення двоетапного алгоритму не значно, але певною мірою підвищує точність розпізнавання емоцій.

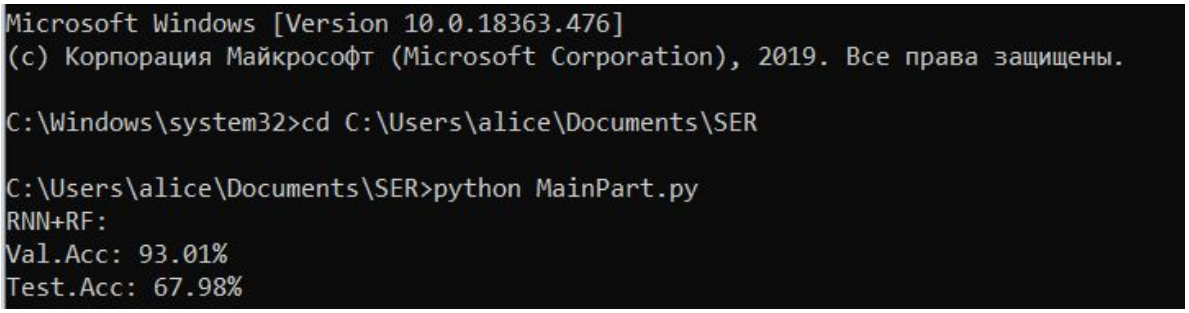

Рисунок 3.2 - Результати роботи двоетапного алгоритму

Видно, що застосування другого класифікатора підвищило точність на декілька відсотків (чого ми і начагалися досягти).

3.4 Висновки за розділом

В даному розділі розглянуто та обгрунтовано вибір програмної мови та архітектури, на яких базується реалізація алгоритму.

Було розгялуно три потужних класифікатори, проведено якісний аналізефективності всіх трьох моделей на обраному датасеті. Найкращою в ефективності навчання виявилася рекурентна нейронна мережа.

В якості мови програмування було обрано мову Python, а в якості основних бібліотек використано librosa, numpy, soundfile, keras, sklearn, PyTorch.

Також було запропоновано власну модифікацію алгоритму розпізнавання емоцій в мові, яку було реалізовано за допомогою описаних

вище бібліотек. Це полегшило розрахунки, які було проведено при реалізації задачі розпізнавання. Було створено консольний додаток, який дозволяє обрати та виконати розпізнавання емоцій по голосу з обраного аудіофайлу.

Продемонстровано, що розбиття алгоритму класифікації на два етапи - підвищує точність, яку сягає модель.

РОЗДІЛ 4 РЕАЛІЗАЦІЯ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї та технологічний аудит стартап-проекту

Стартап проект, що базується на даній роботі має назву «Система розпізнавання емоцій в мові людини».

Даний розділ описує економічну обгрунтованість імплементації цього проекту. В табл. 4.1 приведено опис ідеї стартап-проекту

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Ідея полягає створенні системи сервісів розпізнавання емоцій в мові людини для різного роду споживачів	Емоційний сканер у транспортних компаніях та диспетчерських службах.	Можливість контролю доступу до виконання службових обов'язків осіб, які перебувають в нестійкому або неадекватному стані. Додаткова перевірка пасажирів авіарейсів у рамках заходів протидії тероризму.
	Розпізнавання емоційного стану людини в рамках спілкування	Покращення ефективності роботизованих систем, особливо в сфері спілкування з клієнтами.
	Використання розпізнавання емоційного стану людини для інтернет-реклами	Підлаштування під емоційний стан людини для більш ефективного підбору індивідуальної реклами.
	Розпізнавання рівня напруження людини в ігровій індустрії	Підлаштування важкості рівня гри в залежності від рівня напруженості гравця

Проведено аналіз технологічної здійсненності проекту. Для реалізації основних частин проекту були підібрані відповідні технології. Результати аналізу наведено в таблиці 4.2.

Таблиця 4.2 – Технологічна здійсненність ідеї проекту

№	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1.	Система розпізнавання емоцій в мові людини для ПК	Goodle Cloud ML Engine	Наявна	Платна, недоступна
2.		Tensorflow	Наявна	Безкоштовна, доступна
Обрана технологія реалізації ідеї проекту: для створення системи розпізнавання емоцій в мові використовуються наступні технології: Tensorflow library.				

4.2 Аналіз ринкових можливостей

Перед реалізацією проекта необхідно спланувати напрямки його розвитку враховуючи стан ринкового середовища, потреби споживачів та пропозиції конкурентів. Для цього необхідно визначити ринкові можливості, що можуть бути використані під час ринкового вровадження проекту, а також ринкові загрози, що можуть бути бути перешкодами реалізації проекту.

Рентабельність — поняття, що характеризує економічну ефективність виробництва, за якої за рахунок грошової виручки від реалізації продукції (робіт, послуг) повністю відшкодовує витрати на її виробництво й одержується прибуток як го. Значення середньої рентабельності визначено за формулою:

$$R = \frac{P}{I \times n} \times 100, \quad (4.1)$$

де P - прибуток за час експлуатації проекту;

I - повна сума інвестиційних витрат;

n - час експлуатації проекту.

В таб. 4.3 - попередня характеристика ринку стартап-проекту.

Таблиця 4.3 – Попередня характеристика ринку стартап-проекту

№	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	1000000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	(1000000 * 100) / (250000 * 12) = 33%

Інвестувати грошові засоби доцільно тоді, коли від цього можна отримати більший прибуток, ніж від їх зберігання у банку. Порівнюючи середньорічну рентабельність інвестицій зі ставкою банківського відсотка, можна дійти висновку, що вигідніше.

На таблиці 4.4 наведена характеристика потенційних клієнтів стартап-проекту.

Таблиця 4.4 – Характеристика потенційних клієнтів стартап-проекту

№	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1.	Потреба в автоматичному розпізнаванні емоцій мови людини для швидкого аналізу настрою клієнта.	Сервісні колл-центри; Рекламні агенства	Цільова група зберігає аудіозаписи телефонних розмов у себе в базі. Пристрої можуть мати різну обчислювальну потужність. Користувачі можуть зберігати аудіофайли різного розміру та якості.	Рішення має дозволяти розпізнавати настрій людини за її мовою на аудіо контенті, задовольняючи наступні умови: швидкість реального часу; підтримуватися різними операційними системами. працювати в режимі off line; Автоматичний пошук та групування контенту з подібними емоціями у людей на ньому; інтеграція з внутрішньою БД компанії;

Продовження таблиці 4.4

№	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
2.	Автоматичне розпізнавання негативних емоцій в мові людей з пристроїв запису звуку в публічних місцях (аеропорти, громадський транспорт) в режимі реального часу	Охоронні служби Журналістика та сервіси дуржавного управління та місцевого самоврядування.	Пристрої можуть мати різну обчислювальну потіжність Пристрої можуть мати різну якість звукозапису Пристрої можуть не бути п'єднаними до інтернету	Програмне забезпечення має мати змогу розпізнавати емоції мови в реальному часі: - на різних типах пристроїв (вбудованих пристроях та мобільних пристроях); - можливість аналізувати потокову аудіо трансляцію; - працювати в режимі off-line; - мати програмне та Web API для інтеграції з іншим програмним забезпеченням.
3.	Автоматичне знаходження на аудіо та відео певні мовні емоції та відповідні їм моменти часу на стрічці.	Сервіси для пошуку відео; Користувачі, що займаються монтажем відеороликів	Аудіо, яке супроводжує відеозапис може бути різних розмірів та якості;	Сервіс має генерувати метадані для заданих аудіо та відео файлів про наявні на ньомуемоції та відповідні їм моменти часу на стрічці. Сервіс має опрацьовувати відео різних розмірів та якості Сервіс має здійснювати аналіз прийнятною швидкістю що не перевищує тривалість самого відео.

Наступним кроком є аналіз факторів загроз, що має допомогти передбачити перешкоди на шляху реалізації та впровадження проекту. Фактори загроз наведені в таблиці 4.5.

Таблиця 4.5 – Фактори загроз

№	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1.	Брак фінансування та мала цільова аудиторія на ранніх етапах розвитку проекту	На ринку безліч конкурентів, що мають велику ЦА, рівень фінансування та власну базу даних для аналізу, що дозволить їм швидше виходити на ринок	Активний пошук і залучення інвесторів та партнерів (особливо серед компаній розробників систем управління аудіо контентом) Розширення функціоналу та введення додаткової платної версії продукту.
2.	Конкуренція	Конкурентами є такі компанії, як: Affectiva, EMOSpeech, EmoVoice.	Аналіз слабких місць в фукціональності конкурентів, або тієї ніші на ринку яка слабко покривається ними. Пошук нових сценаріїв застосування розпізнавання емоцій в мові. Інтеграції з іншими програмними системами, що дає нові можливості для користувачів Виділення додаткових інвестицій в удосконалення методів розпізнавання емоцій в мові.

В таблиці 4.6 наведений аналіз факторів можливостей проекту.

Таблиця 4.6 – Фактори можливостей

Фактор	Зміст можливості	Можлива реакція компанії
Інтеграція з хмарними сховищами для відео та аудіо контенту; інтеграція з соціальними мережами за допомогою OAuth 2.0	Користувачі загалом зберігають свої відео та аудіо записи в хмарних сховищах, таких як iCloud, OneDrive, Google Drive тощо. Інтеграція з цими сервісами, а також соціальними мережами (Facebook, Instagram) додасть зручності та доступності цільовій системі, а також розширить коло користувачів	Виділення ресурсів на інтеграцію з усіма популярними хмарними сховищами для зберігання відео та аудіо контенту

Наступним кроком є ступеневий аналіз конкуренції на ринку. Результати аналізу приведені в таблиці 4.7.

Таблиця 4.7 – Ступеневий аналіз конкуренції на ринку.

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Можливі дії компанії, щоб бути конкурентоспроможною*
1. Вказати тип конкуренції - досконала	Існує 3 фірми-конкурентки на ринку	Пошук слабких сторін конкурентів, та альтернативної ніші на ринку Врахувати ціни конкурентних компаній та, використовуючи рекламу, вказати на конкретні переваги перед конкурентами; Пошук шляхів покращення якості продукту
2. За рівнем конкурентної боротьби - міжнародний	Компанії предстваляють різні країни (переважно США)	Використання хмарних послуг для доступності програмного забезпечення в різних куточках світу
3. За галузевою ознакою -- внутрішньогалузева	Конкуренти мають ПЗ, яке використовується лише всередині даної галузі	Розширення функціоналу для вирішення нових задач аналізу аудіо.
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме – програмне забезпечення для розпізнавання емоцій в мові.	Розробити продукт з урахуванням недоліків конкурентів.
5. За характером конкурентних переваг - нецінова - цінова	Користувачу важлива низька ціна на послуги і точність, швидкість розпізнавання	Слід зосередити зусилля на розробку власного програмного забезпечення що реалізовує розпізнавання емоцій з викоистанням безплатних бібліотек і платформ
6. За інтенсивністю - не марочна	Бренди відсутні	-

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі. *М. Портер* вирізняє п‘ять основних факторів, що

впливають на привабливість вибору ринку з огляду на характер конкуренції (див. рис. 4.1).

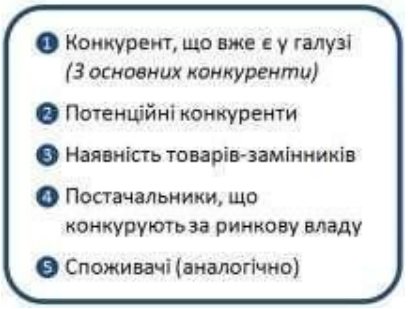

Рисунок 4.1 – Складові моделі 5 сил М. Портера

Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючі їм власні умови співпраці (див. Рис. 4.2).

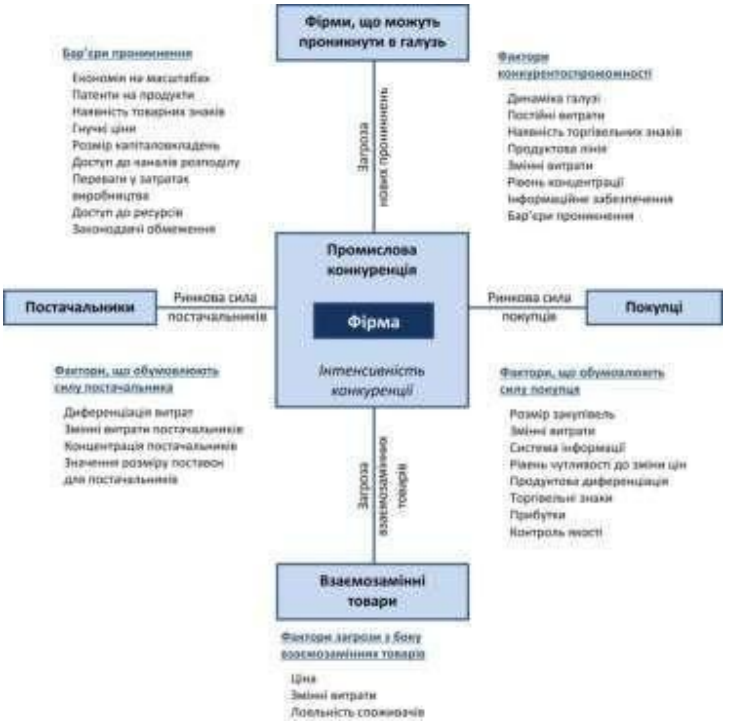

Рисунок 4.2 – Модель 5 сил М. Портера для аналізу конкуренції в галузі

Характеристики факторів моделі відрізняються для різних галузей та змінюються із часом. Сила кожного фактору є функцією від структури галузі та її техніко-економічних характеристик.

На основі аналізу складових моделі 5 сил М. Портера розробляється перелік факторів конкурентоспроможності для певного ринку зображенній на таблиці 4.8

Таблиця 4.8 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари - замінники
	Affectiva, EmoSpeech, EmoVoice	Існуючі системи більш розрекламовані і використовуються іншими content management сервісами	Хмарні послуги для хостингу та масштабування. Хмарна послуга Google Cloud ML Engine. Інтегація з хмарними сховищами через OAuth 2.0	Клієнти диктують свої вимоги до продукту опираючись на законодавство що захищає їх персональні дані.	-
Висновки:	Існує 3 Конкуренти на ринку. Найбільш схожим за виконанням є конкурент 1, оскільки його рішення вже використовується в ігровій індустрії	Шанси виходу на ринок малі. потенційних конкурентів немає, лише прямі.	Постачальник може підняти ціну на свої послуги і цим самим збільшити собівартість послуг	Важливим для користувача є швидкість роботи ПЗ.	-

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспроможності.

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів

маркетингового середовища (табл. 4.6-4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Враховуючи отриманні результати аналіз оформлюється за табл. 4.9.

Таблиця 4.9 – Обґрунтування факторів конкурентоспроможності

№	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1.	Збільшення області використання	Конкуренти використовують послугу точково, коли можна розширити область використання задля розширення кількості цільових клієнтів
2.	Ціновий	Чим менша ціна послуги тим більший попит а отже і вага на ринку. Використання хмарних послуг для реалізації своїх послуг впливає на собівартість продукту. Більші компанії, що мають власну інфраструктуру не мають даної проблеми.

За визначеними факторами конкурентоспроможності проводиться аналіз сильних та слабких сторін стартап-проекту. Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

На таблиці 4.10 описано порівняльний аналіз сильних та слабких сторін проекту.

Таблиця 4.10 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з підприємством*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Великий спектр використання	8				1		2	
2	Цінова політика	10			1,2				
3	Наявний аналіз аудіо та відео контенту	12		1			2		

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза). SWOT-аналіз стартап-проекту представленний у таблиці 4.11

Таблиця 4.11 – SWOT-аналіз стартап-проекту

Сильні сторони: Широке коло споживачів (мобільні пристрої, камери, системи управління аудіо та відео контентом, інші сторонні сервіси що використовують продукт як сервіс для розпізнавання емоцій) Використання ефективних методів та технологій для розпізнавання облич	Слабкі сторони: Слабкий рівень фінансування та мала цільова аудиторія на ранніх етапах. Наявність на ринку сильних конкурентів що уже мають велику цільову аудиторію.
Можливості: Використання хмарних послуг для хостингу. Використання обчислювальних потужностей Google Cloud ML Engine для тренування та виконання моделей розпізнавання емоцій в мові	Загрози: Конкуренція, зміна потреб користувачів, недостатнє фінансування та низький попит

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів. Альтернативи ринкового впровадження стартап-проекту представленні у таблиці 4.12.

Таблиця 4.12 – Альтернативи ринкового впровадження стартап-проекту

№	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1.	З використанням Google Cloud ML Engine	75%	3-5 місяці
2.	Розробка власної інфраструктури для тренування та виконнання моделей розпізнавання емоцій в мові.	25%	7-12 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Тому обираємо альтернативу 1.

4.3 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів що відображається у таблиці 4.13.

Таблиця 4.13 – Вибір цільових груп потенційних споживачів

№	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Користувачі, що зберігають фото та відео контент на своїх пристроях або хмарних сховищах	Користувачі спроможні користуватися даним продуктом і потребують його для зменшення ручної роботи під час управлінням своїм контентом	Користувачі потребують програмне забезпечення для управління своїм відео та фото контентом	Наявні 4 основних і кілька другорядних конкурентів	Поріг входження важкий через конкуренцію, брак фінансування та цільової аудиторії
2.	Компанії що представляють системи управління відео та фото контентом, а також сервісів, що використовують розпізнавання облич для свого функціоналу	Користувачі готові споживати продукт оскільки він доступний як SaaS і вирішує задачі розпізнавання облич за помірну оплату	Розпізнавання облич широко використовується особливо в системах управління відео контентом	Наявні 4 основних і кілька другорядних конкурентів	Поріг входження важкий через конкуренцію, брак фінансування та цільової аудиторії

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. За М. Портером, існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку (за витратами або визначними якостями товару).

Стратегія лідерства по витратах передбачає, що компанія за рахунок чинників внутрішнього і/або зовнішнього середовища може забезпечити більшу, ніж у конкурентів маржу між собівартістю товару і середньоринковою ціною (або ж ціною головного конкурента).

Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів конкурентів. Така відмінність може базуватися на об‘єктивних або суб‘єктивних, відчутних і невідчутних властивостях товару (у ширшому розумінні – комплексі маркетингу), бути реальною або уявною. Інструментом реалізації стратегії диференціації є ринкове позиціонування.

Стратегія спеціалізації передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок. Мета тут полягає в задоволенні потреб вибраного цільового сегменту краще, ніж конкуренти. Така стратегія може спиратися як на диференціацію, так і на лідерство по витратах, або і на те, і на інше, але тільки у рамках цільового сегменту. Проте низька ринкова доля у разі невдалої реалізації стратегії може істотно підірвати конкурентоспроможність компанії.

В таблиці 4.14 приведені деталі базової стратегії розвитку.

Таблиця 4.14 – Визначення базової стратегії розвитку

№	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1.	Створення програми з використанням Google Cloud ML Engine	Ринкове позіціювання	Точність розпізнавання емоцій, пришвидшення роботи, інтеграція з хмарними сховищами та соціальними мережами.	Диференціації

Наступним кроком є вибір стратегії конкурентної поведінки.

Стратегія лідера. Залежно від міри сформованості товарного (галузевого) ринку, характеру конкурентної боротьби компанії-лідери обирають одну з трьох стратегій: розширення первинного попиту, оборонну або наступальну стратегію або ж застосувати демаркетинг або диверсифікацію.

Стратегія розширення первинного попиту доцільна у разі, якщо фірмі-лідерові недоцільно розмінюватися на боротьбу з невеликими конкурентами, вона може отримати велику економічну віддачу від розширення первинного рівня попиту. В цьому випадку компанія займається реалізацією заходів по формуванню попиту(навчанню споживачів користуванню товаром, формування регулярного попиту, збільшення разового споживання), також пропаганду нових напрямів застосувань існуючих товарів, виявлень нових груш споживачів.

У міру зростання ринку, його становлення позиції компанії-новатора починають атакувати конкуренти-імітатори. В цьому випадку, компанія може вибрати оборонну стратегію, метою якої є захист власної ринкової долі.

Наступальна стратегія припускає збільшення своєї частки ринку. При цьому переслідувана мета полягає в подальшому підвищенні прибутковості роботи компанії на ринку за рахунок максимального використання ефекту масштабу.

Якщо фірма потрапляє під дію антимонопольного законодавства, вона може удатися до стратегії демаркетинга, що припускає скорочення своєї частки ринку, зниження рівня попиту на деяких сегментах за рахунок підвищення ціни. При цьому ставиться завдання недопущення на ці сегменти конкурентів, а компенсація втрат прибутку через зменшення обсягів виробництва компенсується встановленням надвисоких цін.

Стратегія виклику лідера. Стратегію виклику лідерові найчастіше вибирають компанії, які є другими, третіми на ринку, але бажають стати лідером ринку. Теоретично, ці компанії можуть прийняти два стратегічні рішення: атакувати лідера у боротьбі за частку ринку або ж йти за лідером.

Рішення атакувати лідера є досить ризикованим. Для його реалізації потрібні значні фінансові витрати, know – how, краще співвідношення «ціна-якість», переваги в системі розподілу і просування і т. д. У разі не реалізації цієї стратегії, компанія може бути відкинута на аутсайдерські позиції на досить довгий час. Залежно від цього компанія може вибрати одну з альтернативних стратегій: фронтальної або флангової атаки.

Стратегія наслідування лідеру. Компанії, що приймають слідування за лідером – це підприємства з невеликою часткою ринку, які вибирають адаптивну лінію поведінки на ринку, усвідомлюють своє місце на нім і йдуть у

фарватері фірм-лідерів. Головна перевага такої стратегії – економія фінансових ресурсів, пов‘язаних з необхідністю розширення товарного(галузевого) ринку, постійними інноваціями, витратами на утримання домінуючого положення.

Стратегія заняття конкурентної ніші. При прийнятті стратегії зайняття конкурентної ніші (інші назви – стратегія фахівця або нішера) компанія в якості цільового ринку вибирає один або декілька ринкових сегментів. Головна особливість – малий розмір сегментів/сегменту. Ця конкурентна стратегія являється похідною від такої базової стратегії компанії, як концентрація. Головне завдання для компаній, що вибирають стратегію нішера або фахівця, – це постійна турбота про підтримку і розвиток своєї конкурентної переваги, формування лояльності і прихильності споживачів, підтримка вхідних бар‘єрів. Визначення базової стратегії конкурентної поведінки представленно у таблиці 4.15

Таблиця 4.15 – Визначення базової стратегії конкурентної поведінки

	Чи є проект «першопрохідцем» на ринку?	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
	Ні	Так	Буде, а саме: Групування контенту за наявністю в ньому певних осіб (конкуренти 2, 3), комбінація різних систем розпізнавання емоцій (в т.ч. розпізнавання емоцій у відеопотоці) для підвищення ефективності системи	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Визначення оптимальної стратегії позиціонування представленно у таблиці 4.16.

Таблиця 4.16 – Визначення стратегії позиціонування

	Вимоги до товару цільової аудиторії	*Базова стратегі я розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
	Точність розпізнавання емоцій в мові, швидкість розпізнавання, інтеграція з хмарними сховищами.	Дифе-ренціації	Використання передових методів та технологій розпізнавання емоцій. Використання Google Cloud ML Engine для тренування та виконання моделей розпізнавання емоцій в мові	Швидкість, точність, доступність з різних хмарних сховищ

4.4 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.17 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

Таблиця 4.17 – Визначення ключових переваг концепції потенційного товару

№	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1.	Висока швидкість розпізнавання емоцій в мові людини	ПЗ працює у розподіленій системі Google Cloud ML Engine, що значно пришвидшує обробку даних. Використання швидкісних моделей розпізнавання облич	Переваги у швидкості обробки контенту
2.	Висока точність розпізнавання	Використання передових архітектур моделей розпізнавання облич, та використання обчислювальних потужностей Google Cloud ML Engine	Переваги в точності розпізнавання емоцій в мові людини

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень. Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості, доставка, умови оплати та ін).

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару. Опис рівнів моделі товару представленний у таблиці 4.18

Таблиця 4.18 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Об‘єкт дає користувачам автоматичне керуваня своїм аудіо та відео контентом в залежності від наявності в ньому певних емоцій. Також надає розпізнавання емоційного стану мови людини в аудіозаписі як сервіс для інших компаній</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Висока точність розпізнавання
2. Висока швидкість роботи
3. Інтеграція з популярними хмарними сховищами даних</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">Якість: згідно до стандарту ISO 4444 буде проведено тестування</td></tr>
<tr><td colspan="3">Маркування відсутнє.</td></tr>
<tr><td colspan="3">Моя компанія. «Emotion Recognition A»</td></tr>
<tr><td rowspan="2">III. Товар із підкріплю-нням</td><td colspan="3">1-місячна пробна безкоштовна версія та безкоштовне
встановлення</td></tr>
<tr><td colspan="3">Постійна підтримка для користувачів</td></tr>
<tr><td colspan="4">Продукт буде з закритим вихідним кодом. Тобто готовий функціонал скопіювати не буде можливості, а захистити ідеї не вдасться.</td></tr>
</table>

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.19). Аналіз проводиться експертним методом. Ціна наведена в колонках 2, 3 та 5 відповідає ціні за обробку 1 гигабайту відео контенту.

Таблиця 4.19 – Визначення меж встановлення ціни

	Рівень цін на товари-замінники	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
	5.5	-	3000000	-

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.20).

Таблиця 4.20 – Формування системи збуту

	Специфіка закупівельної поведінки цільових клієнтів	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
	Базовий функціонал ПЗ для ПК безоплатно	Поширення продукту	0 (напряму), 1 (через одного посередника)	Власна та через посередників
	Отримання розширеного функціоналу для ПК та мобільних пристроїв через підписку зі щомісячною оплатою	Продажа	0 (напряму), 1 (через одного посередника)	Власна та через посередників

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.21)

Таблиця 4.21 – Концепція маркетингових комунікацій

№	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1.	Купівля ПЗ через інтернет, робота з ПЗ на комп‘ютерах з різними ОС, робота з ПЗ на різних типах пристроїв	Інтернет	Швидкодія, простота використання, точність розпізнавання облич	Показати переваги ПЗ, у тому числі і перед конкурентами	Креативна реклама, що показує які можливості надає ПЗ

4.5 Елементи фінансової підтримки стартапу та аналіз ризиків

У таблиці 4.22 наведено структуру інвестиційних витрат на реалізацію стартап-проекту.

Таблиця 4.22 – Сукупні інвестиційні витрати на реалізацію стартап-проекту

№	*Стаття итрат*	*Витрати, тис. грн.*
1.	Початкові витрати	300
1.1.	Проведення пошукових та прикладних досліджень	80
1.2.	Розробка проектних матеріалів і ТЕО	15
1.3.	Проектування	10
1.4.	Придбання обладнання та устаткування та пристроїв	15
1.5.	Придбання нематеріальних активів	50
1.6.	Утримання обладнання та приміщень	30
1.7.	Попередні маркетингові дослідження	20
1.8.	Створення мережі збуту	20
1.9.	Просування та реклами	50
1.10	Юридичні послуги	10
2.	Матеріальні ресурси	0
2.1.	матеріали	0
2.3.	комплектуючі	0
2.4.	сировина	0
3.	Оплату праці команди стартапу	200
Разом		500

4.6 Висновки до розділу

Проведені дослідження показують існування можливості ринкової комерціалізації проекту, оскільки існують потенційні сегменти користувачів. Проте існує висока конкуренція на ринку, що потребує активного аналізу конкурентів та пошуку удосконалення існуючого функціоналу, та пошуку його альтернативного застосування.

Для успішного виконання проекту необхідно реалізувати програму із використанням засобів Goodle Cloud ML Engine. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Згідно з отриманими результатами економічного та рикового аналізу подальша імплементація є доцільною.

ВИСНОВКИ

В даній роботі було описано математичну постановку задачі розпізнавання емоцій за мовою людини. Проведено якісний аналіз існуючих алгоритмів класифікації аудіозаписів за емоційним станом спікера та виявлено найефективніший з них. Найкращим методом класифікації виявилась рекурентна нейромережа із шарами LSTM, оскільки вона має здатність “обирати”, яку інформацію запам’ятовувати протягом більшої кількості кроків, а яку швидко забувати.

Розглянуто ефективно-працюючі ознаки аудіозаписів: мел-частотні кепстральні коефіцієнти, енергія та висота.

Також запропоновано власний алгоритм для вирішення поставленої задачі. Алгоритм базується на введенні двоетапного навчання моделі, що дозволяє враховувати також вплив плину речення на важливість його емоційного стану на цьому етапі. Досягнуто більшої точності, ніж в реалізованому одноетапному алгоритмі, отже використання другого класифікатору у модулі є доцільним у поставленій задачі.

Аргументовано обрання програмного комплексу, за допомогою якого розроблялася реалізація методів класифікації, що представлені в цій роботі. Також надано реалізацію власного алгоритму вирішення задачі класифікації емоцій.

У результаті роботи вдалося підвищити точність розпізнавання емоцій, завдяки розділенню алгоритму класифікації на два етапи.

В роботі використовувались бібліотеки librosa, numpy, soundfile, keras, sklearn, PyTorch мови програмування Python.

Не дивлячись на те, що точність розпізнавання досягнуто доволі високу на рівні існуючих реалізацій, роботу можна ще вдосконалювати. В першу чергу є сенс зосередити увагу на таких можливих вдосконаленнях:

1. Використання більш складних за архітектурою нейронних мереж, наприклад з більшою кількістю шарів та більш складним взаємозв’язком. Набувають популярності згорткові рекурентні НМ, які використовуються і згорткові и рекурентні шари. В поєднанні з застосуванням двоетапного алгоритму, такий підніх може значано підвищити результати.

2. Включення більшої кількості та більш різноманітних акустичних ознак. Це може забезпечити більшу точність навчання, особливо у випадку нейронних мереж.

3. Приділення більш чіткох уваги попоередній обробці аудіозаписів: прибиранню шумів, виділенню високих тонів і т.д.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Automatic Speech Emotion Recognition Using Machine Learning. URL: https://www.intechopen.com/online-first/automatic-speech-emotion-recognition-using-machine-learning
2. P. Sandeep, K. Vaishali A Survey on Emotion Recognition from Speech Signal. *IJARCCE*, Vol. 5, Issue 7, July 2016, P. 11-15.
3. J. Aastha, K. Rajneet A Study of Speech Emotion Recognition Methods. *IJCSMC*, Vol. 2, Issue. 4, April 2013, P. 28-31.
4. Кепстральные коэффициенты как необходимая характеристика процесса создания системы имитации голоса человека с помощью методов глубокого обучения. URL: https://www.eduherald.ru/ru/article/view?id=18125
5. Maximo E. Sanchez–Gutierrez, E. Marcelo Albornoz Deep Learning for Emotional Speech Recognition. *ResearchGate*, Conference Paper, June 2014, P. 23-26.
6. K. Choi, G. Fazekas, M. Sandler, and K. Cho, Convolutional recurrent neural networks for music classification. *arXiv* preprint arXiv:1609.04243, 2016.
7. H. Chengwei, L. Ruiyu, W. Qingyun Practical Speech Emotion Recognition Based on Online Learning: From Acted Data to Elicited Data. *School of Information Science and Engineering*, Southeast University, Nanjing 210096, China, 2013,p. 210.
8. F. Dellaert, T. Polzin, A. Waibel Recognition Emotion in Speech. *School of Computer Science*, Carnegie Mellon University, Pittsburgh, Pennsylvania 152–3890, P. 6-12.

9. Stuhlsatz, A., Meyer, C., Eyben, F., et al., neural networks for acoustic emotion recognition: raising the benchmarks, IEEE Int. *Conf. on Acoustics, Speech and Signal Processing*, P. 5688-5691.
10. Y. Kim and E. Mower Provost, Emotion classication via utterance-level dynamics: A pattern-based approach to characterizing aective expressions, *proceedings of IEEE ICASSP* 2013. *IEEE*, 2013.
11. F. Eyben, M. Wollmer, and B. Schuller, OpenEAR - introducing the Munich open-source emotion and aect recognition toolkit, *in Proceedings of ACII* 2009. IEEE, 2009, p. 16.
12. E. Mower, M. J. Mataric, and S. Narayanan, A framework for automatic human emotion classication using emotion proles, *IEEE Transactions on Audio, Speech, and Language Processing, vol. 19, no. 5*, p. 10571070, 2011
13. Amazon Alexa распознает эмоции по голосу с помощью нейросетей. URL: *https://neurohive.io/ru/novosti/amazon-alexa-raspoznaet-emocii-po-golosu-s-pomoshhju-nejrosetej/*
14. Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). URL: https://doi.org/10.1371/journal.pone.0196391.
15. E. Mower Provost, Identifying salient sub-utterance emotion dynamics using exible units and estimates of aective ow, *in Proceedings of IEEE ICASSP* 2013. IEEE, 2013.
16. Thurid Vogt, Elisabeth Andre Improving Automatic Emotion Recognition from Speech via Gender Differentiation, URL: https://www.informatik.uni-augsburg.de/lehrstuehle/hcm/publications/2006-LREC/
17. Глубокое обучение в задаче распознавания емоций по речи. URL: http://itas2016.iitp.ru/pdf/1570285416.pdf

18. Классификация музыкальных композиций по исполнителям с помощью Скрытых Марковских Моделей. URL: https://habr.com/ru/post/351462/

19. Speech-Emotion-Recognition-Using-Deep-Convolutional. URL: https://www.semanticscholar.org/paper/Speech-Emotion-Recognition-Using-Deep-Convolutional-Zhang-Zhang/2cecfe08fb146110aadd93e6d021fb21f06fda23

20. Добрушкін Г. О., Данилов В. Я., Порівняння якості мел- та барк-частотних кепстральних коефіцієнтів для параметризації мовних сигналів:

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

FeaturesExtractor.py

```
from data_extractor import load_data
import wave
import numpy as np
import math
import os
import pickle
import cf


def weighting(x):
  if weighting_type == 'cos':
    return x * np.sin(np.pi * np.arange(len(x)) / (len(x) - 1))
  elif weighting_type == 'hamming':
    return x * (0.54 - 0.46 * np. cos(2. * np.pi * np.arange(len(x)) / (len(x) - 1)))
  else:
    return x

def calculate_features(frames, freq, options):
  n = len(frames)
  window_sec = 0.2
  window_n = int(freq * window_sec)
  use_derivatives = False

  st_f = cf.stFeatureExtraction(frames, freq, window_n, window_n / 2)
  if st_f.shape[1] > 2:
    i0 = 1
    i1 = st_f.shape[1] - 1
    if i1 - i0 < 1:
      i1 = i0 + 1
    if use_derivatives:
      deriv_st_f = np.zeros((st_f.shape[0]*3, i1 - i0), dtype=float)
    else:
      deriv_st_f = np.zeros((st_f.shape[0], i1 - i0), dtype=float)
    for i in xrange(i0, i1):
      i_left = i - 1
      i_right = i + 1
      deriv_st_f[:st_f.shape[0], i - i0] = st_f[:, i]
      if use_derivatives:
        if st_f.shape[1] >= 2:
          deriv_st_f[st_f.shape[0]:st_f.shape[0]*2, i - i0] = (st_f[:, i_right] - st_f[:, i_left]) / 2.
          deriv_st_f[st_f.shape[0]*2:st_f.shape[0]*3, i - i0] = st_f[:, i] - 0.5*(st_f[:, i_left] + st_f[:, i_right])
    return deriv_st_f
  elif st_f.shape[1] == 2:
```

```
    deriv_st_f = np.zeros((st_f.shape[0], 1), dtype=float)
    deriv_st_f[:st_f.shape[0], 0] = st_f[:, 0]
    if use_derivatives:
      deriv_st_f[st_f.shape[0]:st_f.shape[0]*2, 0] = st_f[:, 1] - st_f[:, 0]
      deriv_st_f[st_f.shape[0]*2:st_f.shape[0]*3, 0] = np.zeros(st_f.shape[0])
    return deriv_st_f
  else:
    deriv_st_f = np.zeros((st_f.shape[0], 1), dtype=float)
    deriv_st_f[:st_f.shape[0], 0] = st_f[:, 0]
    if use_derivatives:
      deriv_st_f[st_f.shape[0]:st_f.shape[0]*2, 0] = np.zeros(st_f.shape[0])
      deriv_st_f[st_f.shape[0]*2:st_f.shape[0]*3, 0] = np.zeros(st_f.shape[0])
    return deriv_st_f
```

CNN.py

```
#This is an TensorFLow implementation of AlexNet by Alex Krizhevsky at all.
import tensorflow as tf
import numpy as np

class AlexNet(object):
    """Implementation of the AlexNet."""

    def __init__(self, x, keep_prob, num_classes, skip_layer,
                 weights_path='DEFAULT'):
        # Parse input arguments into class variables
        self.X = x
        self.NUM_CLASSES = num_classes
        self.KEEP_PROB = keep_prob
        self.SKIP_LAYER = skip_layer

        if weights_path == 'DEFAULT':
            self.WEIGHTS_PATH = 'bvlc_alexnet.npy'
        else:
            self.WEIGHTS_PATH = weights_path

        # Call the create function to build the computational graph of AlexNet
        self.create()

    def create(self):
        conv1 = conv(self.X, 11, 11, 96, 4, 4, padding='VALID', name='conv1')
        norm1 = lrn(conv1, 2, 1e-05, 0.75, name='norm1')
        pool1 = max_pool(norm1, 3, 3, 2, 2, padding='VALID', name='pool1')
        conv2 = conv(pool1, 5, 5, 256, 1, 1, groups=2, name='conv2')
        norm2 = lrn(conv2, 2, 1e-05, 0.75, name='norm2')
        pool2 = max_pool(norm2, 3, 3, 2, 2, padding='VALID', name='pool2')
        conv3 = conv(pool2, 3, 3, 384, 1, 1, name='conv3')
        conv4 = conv(conv3, 3, 3, 384, 1, 1, groups=2, name='conv4')
        conv5 = conv(conv4, 3, 3, 256, 1, 1, groups=2, name='conv5')
        pool5 = max_pool(conv5, 3, 3, 2, 2, padding='VALID', name='pool5')
        flattened = tf.reshape(pool5, [-1, 6 * 6 * 256])
        fc6 = fc(flattened, 6 * 6 * 256, 4096, name='fc6')
        dropout6 = dropout(fc6, self.KEEP_PROB)
        fc7 = fc(dropout6, 4096, 4096, name='fc7')
        dropout7 = dropout(fc7, self.KEEP_PROB)
        self.fc8 = fc(dropout7, 4096, self.NUM_CLASSES, relu=False, name='fc8')

    def load_initial_weights(self, session):
```

```
        weights_dict = np.load(self.WEIGHTS_PATH, encoding='bytes').item()
        for op_name in weights_dict:
            if op_name not in self.SKIP_LAYER:
                with tf.variable_scope(op_name, reuse=True):
                    for data in weights_dict[op_name]:
                        if len(data.shape) == 1:
                            var = tf.get_variable('biases', trainable=False)
                            session.run(var.assign(data))
                        else:
                            var = tf.get_variable('weights', trainable=False)
                            session.run(var.assign(data))


def conv(x, filter_height, filter_width, num_filters, stride_y, stride_x, name,
         padding='SAME', groups=1):


    input_channels = int(x.get_shape()[-1])

    # Create lambda function for the convolution
    convolve = lambda i, k: tf.nn.conv2d(i, k,
                                         strides=[1, stride_y, stride_x, 1],
                                         padding=padding)

    with tf.variable_scope(name) as scope:
        # Create tf variables for the weights and biases of the conv layer
        weights = tf.get_variable('weights', shape=[filter_height,
                                                    filter_width,
                                                    input_channels / groups,
                                                    num_filters])
        biases = tf.get_variable('biases', shape=[num_filters])

    if groups == 1:
        conv = convolve(x, weights)

    # In the cases of multiple groups, split inputs & weights and
    else:
        # Split input and weights and convolve them separately
        input_groups = tf.split(axis=3, num_or_size_splits=groups, value=x)
        weight_groups = tf.split(axis=3, num_or_size_splits=groups,
                                 value=weights)
        output_groups = [convolve(i, k) for i, k in zip(input_groups, weight_groups)]

        conv = tf.concat(axis=3, values=output_groups)
    bias = tf.reshape(tf.nn.bias_add(conv, biases), tf.shape(conv))

    relu = tf.nn.relu(bias, name=scope.name)

    return relu


def fc(x, num_in, num_out, name, relu=True):
    with tf.variable_scope(name) as scope:

            weights = tf.get_variable('weights', shape=[num_in, num_out],
                                      trainable=True)
        biases = tf.get_variable('biases', [num_out], trainable=True)

            act = tf.nn.xw_plus_b(x, weights, biases, name=scope.name)
```

```
    if relu:
        relu = tf.nn.relu(act)
        return relu
    else:
        return act


def max_pool(x, filter_height, filter_width, stride_y, stride_x, name,
             padding='SAME'):
    return tf.nn.max_pool(x, ksize=[1, filter_height, filter_width, 1],
                          strides=[1, stride_y, stride_x, 1],
                          padding=padding, name=name)


def lrn(x, radius, alpha, beta, name, bias=1.0):
    return tf.nn.local_response_normalization(x, depth_radius=radius,
                                              alpha=alpha, beta=beta,
                                              bias=bias, name=name)

def dropout(x, keep_prob):
    return tf.nn.dropout(x, keep_prob)
```

RNN.py

```
import wave
import numpy as np
import math
import os
import pickle
from keras.preprocessing import sequence
from sklearn.ensemble import RandomForestRegressor as RFR

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import LSTM
from keras.optimizers import SGD, Adam, RMSprop
from keras.utils import np_utils
from keras.regularizers import l2
from keras.optimizers import SGD

import h5py

def train_rfr(x, y, tx, ty):
  rfr = RFR(n_estimators=50)
  model = rfr.fit(x, y)
  return model
def train(x, y, options):
  classes = np.unique(y)
  print (classes
  classifiers = []
  for c in classes:
    out = np.array([int(i) for i in y==c])
    print (c, len(out[out == 1]), len(out[out == 0]), len(out[out == 1]) + len(out[out == 0]))
    classifier = RFC(n_estimators=10)
    classifier.fit(x, out)
    classifiers.append(classifier)
  return classifiers
```

```
def train_rfc(x, y, options):
  classifier = RFC(n_estimators=100)

  return classifier.fit(x, y)

def fork (model, n=2):
    forks = []
    for i in range(n):
        f = Sequential()
        f.add (model)
        forks.append(f)
    return forks
def make_sample_lstm(x, n, y=None, use_y=False):
  if use_y:
    xt = np.zeros((x.shape[0], n, x.shape[1] + 1), dtype=float)
    t = np.zeros((x.shape[0], x.shape[1] + 1), dtype=float)
    for i in xrange(x.shape[0]):
      if i == 0:
        t[i, :-1] = x[i, :]
        t[i, -1] = 0
      else:
        t[i, :-1] = x[i, :]
        t[i, -1] = y[i-1]

    for i in xrange(x.shape[0]):
      if i < n:
        i0 = n - i
        xt[i, :i0, :] = np.zeros((i0, x.shape[1] + 1), dtype=float)
        if i > 0:
          xt[i, i0:, :] = t[:i, :]
      else:
        xt[i, :, :] = t[i-n:i, :]
    return xt
  else:
    xt = np.zeros((x.shape[0], n, x.shape[1]), dtype=float)
    for i in xrange(x.shape[0]):
      if i < n:
        i0 = n - i
        xt[i, :i0, :] = np.zeros((i0, x.shape[1]), dtype=float)
        if i > 0:
          xt[i, i0:, :] = x[:i, :]
      else:
        xt[i, :, :] = x[i-n:i, :]
    return xt
class modelLSTM:
  def __init__(self, model, length, use_y):
    self.model = model
    self.n = length
    self.use_y = use_y
  def predict(self, x):
    if self.use_y:
      result = np.zeros((x.shape[0], 1), dtype=float)
      for i in xrange(x.shape[0]):
        t = np.zeros((self.n, x.shape[1] + 1), dtype=float)


    else:
      xt = make_sample_lstm(x, self.n)
      return self.model.predict(xt)
```

```
  def save(self, name_json, name_weights):
    json_string = self.model.to_json()
    open(name_json, 'w').write(json_string)
    self.model.save_weights(name_weights)

def train_lstm_avec(x, y, xt, yt):
  length = 25
  use_y = False

  x_series_train = make_sample_lstm(x, length, y, use_y)
  print (x_series_train.shape)
  x_series_test = make_sample_lstm(xt, length, yt, use_y)
  print (x_series_test.shape)

  print (y[:100, 0])
  model = Sequential()
  model.add(LSTM(256, return_sequences=True, input_shape=(x_series_train.shape[1], x_series_train.shape[2])))
  model.add(Dropout(0.2))
  model.add(Activation('tanh'))
  model.add(LSTM(128))
  model.add(Dropout(0.2))
  model.add(Activation('tanh'))
  # model.add(Dense(128))
  # model.add(Activation('tanh'))
  model.add(Dense(1))
  #model.add(Activation('softmax'))
  model.summary()
  model.compile(loss='mean_absolute_error', optimizer='rmsprop')
  model.fit(x_series_train, y, batch_size=512, nb_epoch=50,
                verbose=2, validation_data=(x_series_test, yt))

  return modelLSTM(model, length, use_y)

def train_lstm(x, y, xt, yt):
  batch_size = 320
  nb_classes = 10
  nb_epoch = 25
  ts = x[0].shape[0]
  model = Sequential()
  model.add(LSTM(512, return_sequences=True, input_shape=(ts, x[0].shape[1])))
  model.add(Activation('tanh'))
  # model.add(LSTM(512, return_sequences=True))
  # model.add(Activation('tanh'))
  model.add(LSTM(256, return_sequences=False))
  model.add(Activation('tanh'))
  model.add(Dense(512))
  model.add(Activation('tanh'))
  model.add(Dense(y.shape[1]))
  model.add(Activation('softmax'))
  model.summary()
  model.compile(loss='categorical_crossentropy', optimizer=RMSprop())
  model.fit(x, y, batch_size=batch_size, nb_epoch=nb_epoch,
                verbose=2, validation_data=(xt, yt), show_accuracy=True)
  return model

def train_mpc(x, y, tx, ty):

  batch_size = 256
  nb_classes = 10
```

```
  nb_epoch = 20

  model = Sequential()
  model.add(Dense(512, input_shape=(x.shape[1],)))
  model.add(Activation('relu'))
  model.add(Dropout(0.2))
  model.add(Dense(1024))
  model.add(Activation('relu'))
  model.add(Dense(1024))
  model.add(Activation('relu'))
  model.add(Dropout(0.2))
  model.add(Dense(512))
  model.add(Activation('relu'))
  model.add(Dense(1024))
  model.add(Activation('relu'))
  model.add(Dense(512))
  model.add(Activation('relu'))
  model.add(Dense(y.shape[1]))
  model.compile(loss='mean_absolute_error',
          optimizer=RMSprop())
  history = model.fit(x, y,
              batch_size=batch_size, nb_epoch=nb_epoch,
              verbose=0, validation_data=(tx, ty), show_accuracy=True)
  return model

def validate1(classifier, test_x, test_y):
  predictions = classifier.predict(test_x)

  total_acc = predictions[predictions == test_y].shape[0] / float(predictions.shape[0])
  print ('Total acc ', total_acc)

  classes = np.unique(test_y)
  print (test_y)
  print (predictions)
  ans = []
  for ci, c in enumerate(classes):
    print (ci)

    idx = np.array([ii for ii, i in enumerate(test_y==c)])
    out = test_y[idx]
    pred = predictions[idx]

    tt = pred[(pred == c) * (out == c)].shape[0] / float(out[out == c].shape[0])
    tf = pred[(pred == c) * (out != c)].shape[0] / float(out[out != c].shape[0])
    ft = pred[(pred != c) * (out == c)].shape[0] / float(out[out == c].shape[0])
    ff = pred[(pred != c) * (out != c)].shape[0] / float(out[out != c].shape[0])
    print (tt, tf, ft, ff, '\t', out[out == c].shape[0] / float(out.shape[0]), out[out != c].shape[0] / float(out.shape[0]))
    tt_tf_ft_ff = [tt, tf, ft, ff]
    ans.append(tt_tf_ft_ff)

  ans_matrix = np.zeros((len(classes), len(classes)), dtype=float)
  for ci, c in enumerate(classes):
    for ci2, c2 in enumerate(classes):
      ans_matrix[ci2, ci] = pred[(pred == c) * (test_y == c2)].shape[0] / float(test_y[test_y == c2].shape[0])
  return np.array(ans, dtype=float), ans_matrix

class Random:
  def __init__(self):
    1
```

```
  def fit(self, x, y):
    1
  def predict(self, x):
    return np.random.normal(0, 0.1, size=x.shape[0])
def train_rnd(x, y, tx, ty):
  return Random()
def validate(classifiers, test_x, test_y):
  print (len(classifiers))
  classes = np.unique(test_y)
  ans = []
  for ci, c in enumerate(classes):
    print (ci)
    out = np.array([int(i) for i in test_y==c])
    predictions = classifiers[ci].predict(test_x)
    tt = predictions[(predictions == 1) * (out == 1)].shape[0] / float(out[out == 1].shape[0])
    tf = predictions[(predictions == 1) * (out == 0)].shape[0] / float(out[out == 0].shape[0])
    ft = predictions[(predictions == 0) * (out == 1)].shape[0] / float(out[out == 1].shape[0])
    ff = predictions[(predictions == 0) * (out == 0)].shape[0] / float(out[out == 0].shape[0])
    print (tt, tf, ft, ff, '\t', out[out == 1].shape[0] / float(out.shape[0]), out[out == 0].shape[0] / float(out.shape[0]))
    tt_tf_ft_ff = [tt, tf, ft, ff]
    ans.append(tt_tf_ft_ff)

  return np.array(ans, dtype=float)
```

HMM.py

```
import os
from hmmlearn import hmm
from sklearn.externals import joblib
import numpy as np
import warnings
warnings.simplefilter('ignore', DeprecationWarning)


class HMMParams:
    def __init__(self, n_components=4, cov_type='diag', n_iter=1000, tol=1e-4):
        self.n_components = n_components
        self.cov_type = cov_type
        self.n_iter = n_iter
        self.tol = tol


class HMMTrainer:

    def __init__(self, hmmParams=HMMParams()):

        self._hmm = hmm.GaussianHMM(n_components=hmmParams.n_components,
                                    covariance_type=hmmParams.cov_type, n_iter=hmmParams.n_iter, tol=hmmParams.tol)

    def train(self, X):
        np.seterr(all='ignore')
        self._hmm.fit(X)

    def get_score(self, input_data):
        return self._hmm.score(input_data)

    def get_score_samples(self, input_data):
        return self._hmm.score_samples(input_data)

    def get_monitorInfo(self):
```

```
        return self._hmm.monitor_

    def save(self, folder_name, class_name, debug_mode=False):

        if folder_name[-1] != '/':
            folder_name += '/'

        filename = folder_name + class_name + '.pkl'

        if debug_mode:
            print('Start saving to ' + filename)

        joblib.dump(self._hmm, filename)

        if debug_mode:
            print('Saving completed ' + filename)

    def load(self, folder_name, class_name, debug_mode=False):

        for filename in [x for x in os.listdir(folder_name) if (x.endswith('.pkl') and (class_name in x))]:
            filepath = os.path.join(folder_name, filename)
            if debug_mode:
                print('Start loading  ' + filename)

            self._hmm = joblib.load(filepath)

            if debug_mode:
                print('Loading completed ' + filename)
```

TrainModels.py

```
import os
import numpy as np
from scipy.io import wavfile
from hmm_trainer import *
from features_calculator import FeauturesCalculator


class TrainHolder:
    def __init__(self, n_components=4, cov_type='diag', n_iter=1000, nfft=512, nmfcc=20, save_folder=None):
        self._hmmParams = HMMParams(n_components, cov_type, n_iter)
        self._mfccCalculator = FeauturesCalculator(nfft=nfft, nmfcc=nmfcc)

        self._models = []
        self.save_folder = save_folder

    def train(self, dataFolder, shuffle=True, seed=None, debug_mode=False):

        np.random.seed(seed)

        if debug_mode:
            print('Current folder is ' + dataFolder)

        lstdirs = [os.path.join(dataFolder, dirname) for dirname in os.listdir(
            dataFolder) if os.path.isdir(os.path.join(dataFolder, dirname))]

        noSubFolders = not lstdirs
        if noSubFolders:
            lstdirs.append(dataFolder)
```

```
    for subfolder in lstdirs:
        if not os.path.isdir(subfolder):
            continue

        label = subfolder[subfolder.rfind('/') + 1:]
        featureMatrix = np.array([])

        for filename in [x for x in os.listdir(subfolder) if x.endswith('.wav')]:
            filepath = os.path.join(subfolder, filename)

            if debug_mode:
                print('Current file is ' + filename)

            features = self.getFeaturesFromWAV(filepath)
            featureMatrix = np.append(featureMatrix, features, axis=0) if len(
                featureMatrix) != 0 else features

        if debug_mode:
            print(subfolder + ' training is started!')

        hmm_trainer = HMMTrainer(hmmParams=self._hmmParams)

        np.random.shuffle(featureMatrix)
        hmm_trainer.train(featureMatrix)

        self._models.append((hmm_trainer, label))

        if self.save_folder is not None:
            hmm_trainer.save(self.save_folder, label, debug_mode)

        if debug_mode:
            print(subfolder + ' training is completed!')
            self._debConvergeInfo(hmm_trainer, label)

    if noSubFolders and debug_mode:
        print(dataFolder + ' training is completed!')

def test(self, folder_name, result_file=None, table_score=False, debug_mode=False):
    if debug_mode:
        print('Tests are started!')

    resultStr = ""
    for input_file in [x for x in os.listdir(folder_name) if x.endswith('.wav')]:
        if debug_mode:
            print('Current file is ' + input_file)

        filepath = os.path.join(folder_name, input_file)

        resultStr += self.testFile(filepath, table_score=table_score)

    if debug_mode:
        print('Concluded!\n Printing results...\n')

    # if result_file = None => print to cmd
    if result_file is None:
        print(resultStr)
    else:
        with open(result_file, 'w') as f:
            f.write(resultStr)
```

```
        if debug_mode:
            print('Results are printed!')

    def testFile(self, filepath, result_file=None, table_score=False):
        """
        filepath : str
            path to song for predict

        result_file : str
            path to file for logs. if is None logs are printing to cmd

        table_score : bool
            prints scores of every model on current song

        returns string with logs
        """
        res = ""

        features = self.getFeaturesFromWAV(filepath)

        scores = {}
        for hmm_model, label in self._models:
            score = hmm_model.get_score(features)
            scores[label] = score
            if table_score:
                res += '    Score ' + str(score) + ' for ' + label + '\n'

        similarity = sorted(scores.items(), key=lambda t: t[1], reverse=True)
        i = 0
        for item in similarity:
            i += 1
            res += str(i) + ' ' + item[0] + '\n'

        # Print the output
        res += os.path.basename(filepath) + ' to ' + \
            str(similarity[0][0]) + '\n'

        if result_file is None:
            print(res)
        else:
            with open(result_file, 'w') as f:
                f.write(res)

        return res

    def save(self, folder_name, debug_mode=False):
        for model, label in self._models:
            model.save(folder_name, label, debug_mode)

    def load(self, folder_name, class_names, debug_mode=False):
        for label in class_names:
            model = HMMTrainer()
            model.load(folder_name, label, debug_mode)
            self._models.append((model, label))

    def getFeaturesFromWAV(self, filename):
        return self._mfccCalculator.getFeaturesFromWAV(filename)
```

```
    def _debConvergeInfo(self, model, label):
        print(label)
        print(model.get_monitorInfo())
        print('\n')

MainPart.py
import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import wave
import numpy as np
import math
import os
import pickle
import models
import csv

import calculate_features as cf
import models

np.random.seed(200)

regime = 'aibo5'

def get_params(regime):
  if regime == 'iemocap':
    #available_emotions = ['ang', 'exc', 'fru', 'neu', 'sad']
    available_emotions = ['ang', 'exc', 'neu', 'sad']
    path_to_samples = 'iem_samples/'
    conf_matrix_prefix = 'iemocap'
    framerate = 44100
    return available_emotions, '', '', '', path_to_samples, conf_matrix_prefix, framerate, '', 0
  elif regime == 'aibo4':
    available_emotions = ['N', 'M', 'E', 'A']

    path_to_wav = 'aibo_data/wav/'
    path_to_transcription = 'aibo_data/transliteration/'
    path_to_labels = 'aibo_data/labels/CEICES/'
    path_to_samples = 'yan_samples_lstm4/'
    conf_matrix_prefix = 'aibo4'
    labels_file = 'word_labels_4cl_aibo_word_set.txt'
    label_pos = 2
    framerate = 16000
    return available_emotions, path_to_wav, path_to_transcription, path_to_labels, path_to_samples,
conf_matrix_prefix, framerate, labels_file, label_pos
  else:
    available_emotions = ['N', 'R', 'E', 'A', 'P']

    path_to_wav = 'aibo_data/wav/'
    path_to_transcription = 'aibo_data/transliteration/'
    path_to_labels = 'aibo_data/labels/IS2009EmotionChallenge/'
    path_to_samples = 'yan_samples_lstm5/'
    conf_matrix_prefix = 'aibo5'
    labels_file = 'chunk_labels_5cl_corpus.txt'
    framerate = 16000
    label_pos = 1
```

```
    return available_emotions, path_to_wav, path_to_transcription, path_to_labels, path_to_samples,
conf_matrix_prefix, framerate, labels_file, label_pos

available_emotions, path_to_wav, path_to_transcription, path_to_labels, path_to_samples, conf_matrix_prefix,
framerate, labels_file, label_pos = get_params(regime)

segmentation = 'by_phrase'

types = {1: np.int8, 2: np.int16, 4: np.int32}

def open_wav(path_to_wav, filename):
  wav = wave.open(path_to_wav + filename, mode="r")
  (nchannels, sampwidth, framerate, nframes, comptype, compname) = wav.getparams()
  content = wav.readframes(nframes)

  samples = np.fromstring(content, dtype=types[sampwidth])
  return (nchannels, sampwidth, framerate, nframes, comptype, compname), samples

def read_lines(f):
  lines = open(f, 'r').read()
  return np.array(lines.split('\n'))


def get_needed_lines(lines, i, subline):
  result = []
  j = i
  condition = True
  while condition:
    if lines[j][:len(subline)] == subline:
      result.append(lines[j])
      j += 1
    else:
      condition = False
  return result, j

def get_label(expert_grades):
  u = np.unique(expert_grades)
  grade = u[0]
  count = expert_grades[expert_grades == grade].shape[0]
  for ui in u:
    current_count = expert_grades[expert_grades == ui].shape[0]
    if current_count > count:
      grade = ui
      count = current_count
  return grade



def read_aibo_data():
  data = []
  files = np.sort([f[:-4] + '.wav' for f in os.listdir(path_to_wav)])
  transliterations = read_lines(path_to_transcription + 'transliteration.txt')
  labels = read_lines(path_to_labels + labels_file)
  index_t = 0
  index_l = 0

  c = 0
  for fi, f in enumerate(files):
    d = {}
```

```
    if fi % 1000 == 0:
      print (f, fi, ' out of ', len(files))
    w = open_wav(path_to_wav, f)
    signal = w[1]
    length = w[0][3] / float(w[0][2])
    t, index_t = get_needed_lines(transliterations, index_t, f[:-4])
    l, index_l = get_needed_lines(labels, index_l, f[:-4])
    if l != []:
      grades = []

      em = l[0].split(' ')[label_pos][0]
      d['id'] = f[:-4]
      d['emotion'] = em
      d['signal'] = signal
      d['transcription'] = t[0][14:-2]
      d['length'] = length
      if (d['emotion'] in available_emotions) and (d['length'] > 0.8):
        data.append(d)
    else:
      c += 1

  print ('missed: ', c)

  return data


sessions = ['Session1', 'Session2', 'Session3', 'Session4', 'Session5']

def get_transcriptions(path_to_transcriptions, filename):
  f = open(path_to_transcriptions + filename, 'r').read()
  f = np.array(f.split('\n'))
  transcription = {}

  for i in xrange(len(f) - 1):
    g = f[i]
    i1 = g.find(': ')
    i0 = g.find(' [')
    ind_id = g[:i0]
    ind_ts = g[i1+2:]
    transcription[ind_id] = ind_ts
  return transcription

def split_wav(wav, emotions):
  (nchannels, sampwidth, framerate, nframes, comptype, compname), samples = wav
  duration = nframes / framerate

  left = samples[0::nchannels]
  right = samples[1::nchannels]

  frames = []
  for ie, e in enumerate(emotions):
    start = e['start']
    end = e['end']

    e['right'] = right[int(start * framerate):int(end * framerate)]
    e['left'] = left[int(start * framerate):int(end * framerate)]

    frames.append({'left':e['left'], 'right': e['right']})
  return frames
```

```
def get_emotions(path_to_emotions, filename):
  f = open(path_to_emotions + filename, 'r').read()
  # print f
  # print f.split('\n')
  f = np.array(f.split('\n'))#np.append(np.array(["]), np.array(f.split('\n')))
  c = 0
  idx = f == "
  idx_n = np.arange(len(f))[idx]
  emotion = []
  for i in xrange(len(idx_n) - 2):
    g = f[idx_n[i]+1:idx_n[i+1]]
    head = g[0]
    i0 = head.find(' - ')
    start_time = float(head[head.find('[') + 1:head.find(' - ')])
    end_time = float(head[head.find(' - ') + 3:head.find(']')])
    actor_id = head[head.find(filename[:-4]) + len(filename[:-4]) + 1:head.find(filename[:-4]) + len(filename[:-4]) + 5]
    emo = head[head.find('\t[') - 3:head.find('\t[')]
    vad = head[head.find('\t[') + 1:]

    v = float(vad[1:7])
    a = float(vad[9:15])
    d = float(vad[17:23])

    emotion.append({'start':start_time,
                    'end':end_time,
                    'id':filename[:-4] + '_' + actor_id,
                    'v':v,
                    'a':a,
                    'd':d,
                    'emotion':emo})
  return emotion

def read_iemocap_data():
  data = []
  for session in sessions:
    print (session)
    path_to_wav = 'iemocap_data/' + session + '/dialog/wav/'
    path_to_emotions = 'iemocap_data/' + session + '/dialog/EmoEvaluation/'
    path_to_transcriptions = 'iemocap_data/' + session + '/dialog/transcriptions/'

    files = os.listdir(path_to_wav)
    files = [f[:-4] for f in files]
    print (len(files))
    print (files)
    for f in files:
      emotions = get_emotions(path_to_emotions, f + '.txt')
      wav = open_wav(path_to_wav, f + '.wav')
      sample = split_wav(wav, emotions)

      transcriptions = get_transcriptions(path_to_transcriptions, f + '.txt')
      for ie, e in enumerate(emotions):
        if e['emotion'] in available_emotions:
          e['signal'] = sample[ie]['left']
          #e['right'] = sample[ie]['right']
          e['transcription'] = transcriptions[e['id']]
          data.append(e)
  return data
```

```
def read_data():
  if regime == 'aibo4' or regime == 'aibo5':
    return read_aibo_data()
  else:
    return read_iemocap_data()

def check_all_finite(X):
  X = np.asanyarray(X)
  if (X.dtype.char in np.typecodes['AllFloat'] and not np.isfinite(X.sum())
        and not np.isfinite(X).all()):
    return True
  else:
    return False

def to_categorical(y):
  y_cat = np.zeros((len(y), len(available_emotions)), dtype=int)
  for i in xrange(len(y)):
    y_cat[i, :] = np.array(np.array(available_emotions) == y[i], dtype=int)

  return y_cat

def save_sample(x, y, name):
  with open(name, 'w') as csvfile:
    w = csv.writer(csvfile, delimiter=',')
    for i in xrange(x.shape[0]):
      row = x[i, :].tolist()
      row.append(y[i])
      w.writerow(row)

def load(name):
  with open(name, 'r') as csvfile:
    r = csv.reader(csvfile, delimiter=',')
    x = []
    y = []
    for row in r:
      x.append(row[:-1])
      y.append(row[-1])
  return np.array(x, dtype=float), np.array(y)

def get_features(data, save=True, path=path_to_samples):
  failed_samples = []
  for di, d in enumerate(data):
    if di%1000 == 0:
      print (di, ' out of ', len(data))
    st_features = cf.calculate_features(d['signal'], framerate, None).T
    x = []
    y = []
    for f in st_features:
      if f[1] > 1.e-4:
        x.append(f)
        y.append(d['emotion'])
    x = np.array(x, dtype=float)
    y = np.array(y)

    if save:
      save_sample(x, y, path + d['id'] + '.csv')
  return x, y

def get_field(data, key):
```

```
  return np.array([e[key] for e in data])


def balance_sample(x, y, size):
  labels = np.unique(y)
  xc = {}
  yc = {}
  for l in labels:
    xc[l] = x[y == l]
    yc[l] = y[y == l]


  s = size / len(labels)

  tx = np.zeros((s*len(labels), x.shape[1]), dtype=float)
  ty = np.zeros(s*len(labels), dtype=str)
  for i in xrange(len(labels)):
    j = i*s
    n = xc[labels[i]].shape[0]

    idx = np.random.randint(low=0, high=n, size=s)
    tx[j:j+s, :] = xc[labels[i]][idx, :]
    ty[j:j+s] = yc[labels[i]][idx]

  return tx, ty

def get_sample(idx, path):
  tx = []
  ty = []
  for i in idx:
    x, y = load(path + '/' + i + '.csv')
    if len(x) < 40:
      tx.append(np.array(x, dtype=float))
      #tx.append(np.array(x, dtype=float))
      ty.append(y[0])

  tx = np.array(tx)
  ty = np.array(ty)
  return tx, ty

def normalize(x):
  gminx = np.zeros(x[0].shape[1]) + 1.e5
  gmaxx = np.zeros(x[0].shape[1]) - 1.e5
  for i in xrange(x.shape[0]):
    q = x[i]
    minx = np.min(q, axis=0)
    maxx = np.max(q, axis=0)

    for s in xrange(x[0].shape[1]):
      if gminx[s] > minx[s]:
        gminx[s] = minx[s]
      if gmaxx[s] < maxx[s]:
        gmaxx[s] = maxx[s]

  for i in xrange(x.shape[0]):
    for s in xrange(x[0].shape[1]):
      x[i][:, s] = (x[i][:, s] - gminx[s]) / float(gmaxx[s] - gminx[s])
```

```
  return x

def grow_sample(x, y, n=10000):
  xg = []
  yg = []
  eps = 5.*1.e-2
  for i in xrange(n):
    j = np.random.randint(x.shape[0])
    x0 = x[j]
    x0 += eps * np.random.normal(0, 1, size=x0.shape)
    y0 = y[j]
    xg.append(x0)
    yg.append(y0)
  return np.array(xg), np.array(yg)

def reshape_for_dense(x, y):
  j = 0
  xr = []
  yr = []
  for i in xrange(x.shape[0]):
    for k in xrange(x[i].shape[0]):
      xr.append(x[i][k, :])
      yr.append(y[i])
  return np.array(xr), np.array(yr)


def pad_sequence(x, ts):
  xp = []
  for i in xrange(x.shape[0]):
    x0 = np.zeros((ts, x[i].shape[1]), dtype=float)
    if ts > x[i].shape[0]:
      x0[ts - x[i].shape[0]:, :] = x[i]
    else:
      maxe = np.sum(x[i][0:ts, 1])
      for j in xrange(x[i].shape[0] - ts):
        if np.sum(x[i][j:j + ts, 1]) > maxe:
          x0 = x[i][j:j + ts, :]
          maxe = np.sum(x[i][j:j + ts, 1])
    xp.append(x0)
  return np.array(xp)



data = np.array(read_data())
print (data)
print (len(data))

ids = get_field(data, 'id')
emotions = get_field(data, 'emotion')
print (np.unique(emotions))

for i in xrange(len(available_emotions)):
  print (available_emotions[i], emotions[emotions == available_emotions[i]].shape[0])

parts = 5
permutation = np.random.permutation(len(data))
permuted_ids = ids[permutation]

step = len(data) / parts
```

```
preds = []
trues = []

get_features(data)

for part in xrange(parts):
  i0 = step * part
  i1 = step * (part + 1)

  train_idx = np.append(permuted_ids[:i0], permuted_ids[i1:])
  test_idx = permuted_ids[i0:i1]

  train_x, train_y = get_sample(train_idx, path_to_samples)



  # energies = []
  # for i in xrange(train_x.shape[0]):
  #   energies.append(np.mean(train_x[i][:, 1]))

  # quantile = 0.5
  # largest_energy_idx = np.argsort(energies)[:int(quantile*len(energies))]

  # train_x = train_x[largest_energy_idx]
  # train_y = train_y[largest_energy_idx]

  test_x, test_y = get_sample(test_idx, path_to_samples)

  # train_x = normalize(train_x)
  # test_x = normalize(test_x)

  #train_x, train_y = grow_sample(train_x, train_y, 10000)
  #train_x, train_y = reshape_for_dense(train_x, train_y)
  print (train_x.shape)
  print (train_y.shape)

  # lengths = {}
  # for i in xrange(train_x.shape[0]):
  #   if train_x[i].shape[0] in lengths.keys():
  #     lengths[train_x[i].shape[0]] += 1
  #   else:
  #     lengths[train_x[i].shape[0]] = 1

  # for k, v in lengths.items():
  #   print k, v

  ts = 32

  train_x = pad_sequence(train_x, ts)
  test_x = pad_sequence(test_x, ts)

  train_y_cat = to_categorical(train_y)
  test_y_cat = to_categorical(test_y)
  model = models.train_lstm(train_x, train_y_cat, test_x, test_y_cat)

  scores = model.predict(test_x)
  prediction = np.array([available_emotions[np.argmax(t)] for t in scores])
  print (prediction[prediction == test_y].shape[0] / float(prediction.shape[0]))
```

```
  #test_x, test_y = get_sample(train_idx, 'yan_samples_lstm4')

  for i in xrange(len(prediction)):
    preds.append(prediction[i])
    trues.append(test_y[i])

class_to_class_precs = np.zeros((len(available_emotions), len(available_emotions)), dtype=float)

preds = np.array(preds)
trues = np.array(trues)

print ('Total accuracy: ', preds[preds == trues].shape[0] / float(preds.shape[0]))

for cpi, cp in enumerate(available_emotions):
  for cti, ct in enumerate(available_emotions):
    #print cp, ct, emo_pred[(emo_pred == cp) * (emo_test == ct)].shape[0], emo_test[emo_test == ct].shape[0]
    if trues[trues == ct].shape[0] > 0:
      class_to_class_precs[cti, cpi] = preds[(preds == cp) * (trues == ct)].shape[0] / float(trues[trues == ct].shape[0])
    else:
      class_to_class_precs[cti, cpi] = 0.


fig, ax = plt.subplots()
heatmap = ax.pcolor(class_to_class_precs.T, cmap=plt.cm.Blues)

ax.set_xticklabels(available_emotions, minor=False)
ax.set_yticklabels(available_emotions, minor=False)

ax.set_xticks(np.arange(len(available_emotions)) + 0.5, minor=False)
ax.set_yticks(np.arange(len(available_emotions)) + 0.5, minor=False)

for i in xrange(len(available_emotions)):
  for j in xrange(len(available_emotions)):
    plt.text(i+0.2, j+0.4, str(class_to_class_precs[i, j])[:5])

plt.savefig(conf_matrix_prefix + 'lstm.png')
plt.close()
```